Table of Contents

© 2014 Jacek Spolnik

© 2014 Jacek Spolnik

© 2014 Jacek Spolnik

© 2014 Jacek Spolnik

© 2014 Jacek Spolnik

© 2014 Jacek Spolnik

© 2014 Jacek Spolnik

© 2014 Jacek Spolnik

1 Introduction

Welcome to the documentation of NDatabase.

To avoid impedance mismatch overhead between Object and Relational worlds, give a try to NDatabase.

NDatabase is a new generation Object Database: a real native and transparent persistence layer for .NET which is really easy to use.

- **Object** because the basic persistent unit is an object, not a table.
- **Native and Transparent** because it directly persists objects the way they exist in the native programming language, without any conversion.

Project

You can find the project home site here. Additionally, on codeplex you can find project sources, download files, discussions around the project and more.

Missing Documentation?

If you are missing documentation or feel that documentation is out of date, please let us know and that will be fixed!

1.1 Why NDatabase?

Storage - all what you want

NDatabase is focused on storing any object in .NET in just one line. You don't need `[Serializable]` attribute, you don't need any custom interface and you don't need to specify Object ID in your classes. Just save your objects as they are.

You can save object with private fields, then you can query for this fields. Even if the private field is read-only, it will be still stored as the part of the object and you can query for that any time. Circular references, immutable types, Exception class, collections, arrays, nullable types, primitive types and any custom type, all are supported by NDatabase.

Release - care about the details

Every release contains a lot of checks before it sees the light of the day. When features are committed, TeamCity starts the job. It builds the library from the scratch, then it runs all unit tests and integration tests which need to pass before having final version, then several metrics need to have acceptable values: unit tests coverage, number of duplicates, inspections failure number (ReSharper inspections embedded into TeamCity 7), so all things which are important for software high quality.

Finally, existing samples are checked if they are working correctly with the new version of NDatabase, and that's the last yes or no for the release version. With every release you are sure, that delivered product has high quality.

Performance - it is really fast

© 2014 Jacek Spolnik

About 10% - 20% faster than Json.NET, faster than many built-in .NET serializers, at least as fast as db4o and other object databases!

Small effort to start use - big benefit when used

NDatabase is very simple and intuitive: the learning time is very short. Have a look at the NDatabase one minute tutorial to check this.

The API is simple and does not require learning any mapping technique. There is no need for mapping between the native objects and the persistence store.

Using NDatabase as your persistence layer will let you focus on your business logic allowing storing and retrieving native objects in a single line of code. No more Relational to Object mapping is necessary, NDatabase just persists objects the way they are, no matter their complexity.

Looking for more?

Check existing features, NDatabase has much more to offer. If you still missing something, let us know!

1.2 Overview

NDatabase is a new generation Object Oriented Database. It is a real transparent persistence layer that allows anyone to persist native .net objects with a single line of code.

NDatabase can be used as an embedded database engine that can be seamlessly integrated to any product as an embedded database.

Simple

NDatabase is very simple and intuitive: the learning time is very short. Have a look at the NDatabase one minute tutorial to check this. The API is simple and does not require learning any mapping technique. There is no need for mapping between the native objects and the persistence store.

NDatabase simply stores objects the way they are.

NDatabase requires zero administration and zero installation.

Small

The NDatabase library is less than 300kb and is distributed as a single dll that can easily be packaged in any type of .net application.

Safe and Robust

NDatabase supports ACID transactions to guarantee data integrity of the database. All committed work will be applied to the database even in case of hardware failure. This is done by automatic transaction recovery on the next startup.

One single database file

NDatabase uses a single file to store all data:

© 2014 Jacek Spolnik

- The Meta-model
- The objects
- The indexes

Productivity

NDatabase lets you persist data with a very few lines of code. There is no need to modify the classes that must be persisted and no mapping is needed. So developers can concentrate on business logic implementation instead of wasting time with the persistence layer.

Easy to integrate

The only requirement to use NDatabase is to have a single dll on the application path.

Linq support with LinqPad driver

Use Linq as your query language. View your NDatabase data with last version of LinqPad.

Triggers

NDatabase currently supports 4 types of triggers:
- Select trigger (after)
- Insert trigger (before, after)
- Delete trigger (before, after)
- Update trigger (before, after)

Indexes

NDatabase supports field named indexes based on B-Trees which is accessible by NDatabase API.

Refactoring

NDatabase currently supports 4 types of refactoring:
- Adding a new field (via API)
- Removing a field (via API)
- Rename a class (via API)
- Rename a field (via API)

Logging

NDatabase is using widely embedded logger. You can write plug-in to use any external logger
- Sample for log4net is here.

Northwind

Well-known Northwind database available as object database.

License

© 2014 Jacek Spolnik

NDatabase is distributed under the LGPL license.

1.3 How it works

Persist an object with NDatabase

```
// Create the instance be stored
var sport = new Sport("volley-ball");

// Open the database
using (var odb = OdbFactory.Open("test.db"))
{
    // Store the object
    odb.Store(sport);
}
```

Retrieve object from NDatabase

```
// Open the database
using (var odb1 = OdbFactory.Open("test.db"))
{
    var sports = odb1.Query<Sport>();

    // code working on sports list
}
```

The Sport class definition

```
public class Sport
{
    public Sport(string name)
    {
        Name = name;
    }

    public string Name { get; set; }

    public override string ToString()
    {
        return Name;
    }
}
```

Want to see more ?

Check Getting Started.

1.4 Release Notes

NDatabase 3.8

- CascadeDelete attribute - allowing on cascade delete (see issue page) - documentation
- OID attribute - allows on mapping internal OID to class defined field (see issue page) - documentation
- Bug fix - null values of index fields (compare to / equals methods) (see issue page)
- Bug fix - Select trigger should pass object instead of object representation to the select after method

© 2014 Jacek Spolnik

NDatabase 3.7

- Redefine NDatabase structure - changes in namespaces, decoupling classes (see notes) - breaking change
- Values Query - refresh api, bug fixes, add unit tests and new documentation (see page) - breaking change
- Bug fix for UpdateMetaModel Exception on Added Class (see issue page)

NDatabase 3.6

- Add multithreading and multiprocess support to NDatabase (Isolation Level - serializable, see more info)
- NDatabase is strongly signed assembly now

NDatabase 3.5

- Add .net 4.5 version of NDatabase
- Class resolver changed, now there is only one standard. Classes are always identified by assembly qualified name, but if assembly cannot be found for the class, code will try to enumerate all assemblie loaded into runtime to match the class. **(BREAKING CHANGE)**
- All transactions are working in in-memory mode
- Logging engine - simplify it, clean from non portable code, improve performance
- Remove obsolete code invocation Assembly.LoadWithPartialName (http://ndatabase.codeplex.com/workitem/902)
- Move OrderByConstants and NonPersistentAttribute to NDatabase.Common namespace (BREAKING CHANGE)

Additional changes:

- Silverlight, CompactFramework and Windows Phone versions will not be actively developed any more. Last stable version (NDatabase 3.4) you can find in mobile branch, and in master35 branch (CF). Feel free to adapt them to your needs.

NDatabase 3.4

- Performance improvements: Buffer.BlockCopy instead of Array.Copy, simplify logging messages - remove less important messages
- Add FSharp sample project
- Add WCF sample
- Fix issue with hidden exceptions during triggers execution
- Remove console logger from engine, it could be attached on demand (simple class), and it is thing which cannot be moved to other .net platforms (mobile, silverlight)
- Remove TransactionId (that's causing changes in db even when we are only querying for data) - this is not exactly related to transaction engine

NDatabase 3.3

- Two new versions of NDatabase (Silverlight, Windows Phone 7) (see limitations)
- Two new samples covering Silverlight app and Windows Phone app
- Performance Improvements in Internal Engine

© 2014 Jacek Spolnik

- Make EnableNormalTypeResolutionMode internal

NDatabase 3.2

- Portability - implementing less restricted Type resolution - without assembly name (namespace and class name are used)
- Remove bug in Class introspector (it processed only user classes) (internal)
- Remove partition of classes to user classes and system classes. Now all classes are treated in the same way (internal)
- Fix in RefactorManager - rename class refactoring, remove field refactoring (Refactoring API)
- Add unit tests for RefactorManager class (internal)
- Generating less restricted Northwind database regarding Type resolution - Northwind is portable now (Northwind sample db)
- Fix ObjectRepresentation - setting values of object attribute (Triggers API)
- Add unit tests for ObjectRepresentaiton class (internal)
- Add documentation for SODA queries in documentaiton

NDatabase 3.1

- NDatabase as In-Memory Database
- remove app.config files from samples
- New project home page - http://ndatabase.wix.com/home
- New project documentation page - http://ndatabase.net/
- Updating and adding xml documenation which covers all public api
- performance improvements in seeking the file (changing position happen only just before reading or writing)
- minor cleanup (remove unused things, update access levels)
- improve design of Northwind sample - extract typed data context to another library
- move NonPersistentAttribute to Odb.Core namespace
- move IConstraint to Odb.Core.Query namespace
- move IRefactorManager to Odb.Core namespace
- Move all exception or error related classes under NDatabase.Exceptions namespace
- move OIDFactory to Odb.Core namespace
- move IOdbComparable to Odb.Core namespace

NDatabase 3.0

- Query API interface changed to SODA queries
- Linq to NDatabase added (support for Linq, translated to SODA queries)
- LinqPad driver added
- Sample database prepared - northwind.ndb (with generator from Sql Server)
- Collections and dictionaries are now saved as normal objects (they could be stored directly in NDatabase) - breaking db layer change
- Namespaces are again starting with NDatabase (instead of NDatabase2), with the new versions only assembly name will be changed, now it is NDatabase3.dll
- Minor bug fixes
- Signing NDatabase assembly
- NDatabase moved to GIT
- Update DB version to 30

© 2014 Jacek Spolnik

NDatabase 2.2

- Migration to .NET 4
- Add support for unsigned int, short, long and signed byte to NDatabase
- Improve Triggers API (make it similar to Index one) - breaking change
- Remove from `IOdb` interface methods like `AddUpdateTrigger`, `AddDeleteTrigger`, etc.
- Add to `IOdb` interface `ITriggerManager TriggerManagerFor<T>() where T : class` method, which allows on all CRUD operations in terms of triggers
- Changing base NDatabase API to be more developer friendly - breaking change
- Instead of `GetObjects<T>` methods in `IOdb` interface we have `Query<T>` methods
- `IClassRepresentation` replaced with `IIndexManager` which controls access/creation of indexes (simplify the Index API)
- `CriteriaQuery` methods of `IOdb` interface replaced with more meaningful `CreateCriteriaQuery` methods (they are factory methods)
- Change `IOdb` inteface's `GetName` method to `GetDbId` (more meaningful)
- Rename `IObejcts<T>` inteface with `IObjectSet<T>` (more Soda standard here)
- Update access level of NDatabase exception constructors (all are internal)
- Make all internal exceptions the concrete classes of `OdbRuntimeException` (all NDatabase exceptions are public)
- Update db version to 20, earlier db files are not compatible (need to use converter) (db version compatibility runtime check)
- Add Exception Handling Sample
- Add CSV Sample, which is showing how to take advantage when we will process csv file into NDatabase with index (querying data - performance about 4 times better than when using FileHelpers)
- `public static IConstraint InvariantEqual<T>(string attributeName, T value)` method relaced with `public static IConstraint InvariantEqual(string attributeName, string value)`
- Clean `CriteriaQuery` internal code (preparing for LINQ Provider implementation)
- More minor cleanup

NDatabase 2.0

- namespaces, dll name both are changed to NDatabase2
- query API is changed. Now it is using generics in every possible place
- changing the way, how fields from class are stored - for now they are ordered by name which allows db on working well even if someone will change the order of fields in class definition - breaking change
- Added Coverter between NDatabase 1.0.4 and NDatabase 2.0
- Update Logging API, clean code connected with Logging API and logging usage in NDatabase (http://ndatabase.codeplex.com/workitem/802,)
- Add documentation for NDatabase logging
- Make the `ODBRuntimeException` the public class (possibility to catch NDatabase exception for external world), the same for `IError`
- remove boxing in many places (all possible places for now are using underlying types)
- Enum is serialized as the class, not any more as the string
- cleaned `OdbConfiguration`
- fixed bug for indexes on property names
- add new unit tests for Layer 2 and Layer 3
- Fixed minor bug in `ObjectIntrospector` - comparing full class name with simple class name
- Samples updated to NDatabase 2.0.1

NDatabase 1.0.4

© 2014 Jacek Spolnik

- Bug fix: storing of generic Dictionary
- Fix issue connected with total size of string required to write into file
- Simplify the way how string is serialized
- New unit tests for Layer 3 and Layer 2
- Extend NuGet specification for NDatabase project
- Changing many internal classes permission from public to internal

NDatabase 1.0.3

- Decimal is now treated as the native type
- Decimal is binary serializable for now instead of using string serialization
- add new unit tests
- code fixes (replace ArrayList with List<object> when applicable)
- fixed issue with AutoDelete
- Fix issues from code inspections (TeamCity)

NDatabase 1.0.2

- performance improvements, see the page
- couple of new unit tests
- code clean up & unit tests cleanup

NDatabase 1.0.1

- Initial stable version of NDatabase Lightweight C# Object Database

2 Getting Started

The best way to learn how to use NDatabase is to have a look to these two small tutorials:
- 1 Minute Tutorial
- 5 Minutes Tutorial

2.1 1 Minute Tutorial

Here is a simple example to demonstrate how it is easy to persist objects with NDatabase, and then how to use persisted objects.

Here, a C# instance is created, the NDatabase database is opened, the store method is called to save two objects and then the database is closed (auto commit). After that we could use persisted objects, so database is opened again and then we are displaying the content of persisted objects.

To restore instance by class, we simply need to use `Query<T>` method to create a query for `T` type objects, and then we are executing this query. The result will contain all objects derived from mentioned base class or interface.

© 2014 Jacek Spolnik

```
const string dbFileName = "game.db";

const double mageAttackValue = 3.3;
const double mageDefenseValue = 3.4;

const double warriorAttackValue = 4.4;
const double warriorDefenseValue = 2.2;

// create two objects
IHero mage = new Mage("Merlin", mageAttackValue, mageDefenseValue);
IHero warrior = new Warrior("Conan", warriorAttackValue, warriorDefenseValue);

// store them
using (var odb = OdbFactory.Open(dbFileName))
{
    odb.Store(mage);
    odb.Store(warrior);
}

// retrieve them by classes and by interface
using (var odb = OdbFactory.Open(dbFileName))
{
    // work with mages
    var mages = odb.Query<Mage>().Execute<Mage>();
    foreach (var hero in mages)
        Console.WriteLine(hero);

    // work with warriors
    var warriors = odb.Query<Warrior>().Execute<Warrior>();
    foreach (var hero in warriors)
        Console.WriteLine(hero);

    Console.WriteLine("Start working with IHero interface.");

    // work with heroes
    var heroes = odb.Query<IHero>().Execute<IHero>();
    foreach (var hero in heroes)
        Console.WriteLine(hero);
}
```

Output from the tutorial

```
[Mage] Name: Merlin, Attack: 3,3, Defense: 3,4
[Warrior] Name: Conan, Attack: 4,4, Defense: 2,2

Start working with IHero interface.

[Mage] Name: Merlin, Attack: 3,3, Defense: 3,4
[Warrior] Name: Conan, Attack: 4,4, Defense: 2,2
```

The base set of classes used in this sample

Simple interface `IHero` which describes any hero implementation.

© 2014 Jacek Spolnik

```
public interface IHero
{
    string Name { get; }

    double Attack { get; }
    double Defense { get; }
}
```

Mage implementation of `IHero` interface. Represents the `Mage`.

```
public sealed class Mage : IHero
{
    public Mage(string name, double attack, double defense)
    {
        Name = name;
        Attack = attack;
        Defense = defense;
    }

    #region Implementation of IHero

    public string Name { get; private set; }
    public double Attack { get; private set; }
    public double Defense { get; private set; }

    #endregion

    public override string ToString()
    {
        return string.Format("[Mage] Name: {0}, Attack: {1}, Defense: {2}",
                                    Name, Attack, Defense);
    }
}
```

Warrior implementation of `IHero` interface. Represents the `Warrior`.

```
public sealed class Warrior : IHero
{
    public Warrior(string name, double attack, double defense)
    {
        Name = name;
        Attack = attack;
        Defense = defense;
    }

    #region Implementation of IHero

    public string Name { get; private set; }
    public double Attack { get; private set; }
    public double Defense { get; private set; }

    #endregion

    public override string ToString()
    {
        return string.Format("[Warrior] Name: {0}, Attack: {1}, Defense: {2}",
                                    Name, Attack, Defense);
    }
}
```

© 2014 Jacek Spolnik

If you want to learn more about NDatabase, click here to have a look at the 5 minutes tutorial.

2.2 5 Minutes Tutorial

Here are 10 steps examples that demonstrate how to store, query and update objects with NDatabase. Query steps are doubled for SODA and LINQ to NDatabase queries.

Step 1: Create and store a Sport instance

```
var sport = new Sport("volley-ball");

using (var odb = OdbFactory.Open(TutorialDb5MinName))
    odb.Store(sport);
```

Step 2: Create and store Game instance

```
// Create instance
var volleyball = new Sport("volley-ball");

// Create 4 players
var player1 = new Player("julia", DateTime.Now, volleyball);
var player2 = new Player("magdalena", DateTime.Now, volleyball);
var player3 = new Player("jacek", DateTime.Now, volleyball);
var player4 = new Player("michal", DateTime.Now, volleyball);

// Create two teams
var team1 = new Team("Krakow");
var team2 = new Team("Skawina");

// Set players for team1
team1.AddPlayer(player1);
team1.AddPlayer(player2);

// Set players for team2
team2.AddPlayer(player3);
team2.AddPlayer(player4);

// Then create a volley ball game for the two teams
var game = new Game(DateTime.Now, volleyball, team1, team2);

using (var odb = OdbFactory.Open(TutorialDb5MinName))
    odb.Store(game);
```

Step 3 (SODA): Example of a simple query

© 2014 Jacek Spolnik

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var query = odb.Query<Player>();
    query.Descend("Name").Constrain("julia").Equal();
    var players = query.Execute<Player>();

    Console.WriteLine("\nStep 3 (Soda): Players with name julia");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players, Has.Count.EqualTo(1));
}
```

Step 3 (LINQ to NDatabase): Example of a simple query

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var players = from player in odb.AsQueryable<Player>()
                  where player.Name.Equals("julia")
                  select player;

    Console.WriteLine("\nStep 3 (Linq): Players with name julia");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players.Count(), Is.EqualTo(1));
}
```

Step 4 (SODA): Example of query navigating between object relations

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var agassi = new Player("André Agassi", DateTime.Now, new Sport("Tennis"));
    odb.Store(agassi);

    var query = odb.Query<Player>();
    query.Descend("FavoriteSport").Descend("_name")
         .Constrain("volley-ball").Equal();

    var players = query.Execute<Player>();

    Console.WriteLine("\nStep 4 (Soda): Players of Voller-ball");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players, Has.Count.EqualTo(4));
}
```

Step 4 (LINQ to NDatabase): Example of query navigating between object relations

© 2014 Jacek Spolnik

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var players = from player in odb.AsQueryable<Player>()
                  where player.FavoriteSport.Name == "volley-ball"
                  select player;

    Console.WriteLine("\nStep 4 (Linq): Players of Voller-ball");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players.Count(), Is.EqualTo(4));
}
```

Step 5 (SODA): Example of query using object references

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    // retrieve the volley ball sport object
    var query = odb.Query<Sport>();
    query.Descend("_name").Constrain("volley-ball").Equal();
    var volleyBall = query.Execute<Sport>().GetFirst();

    Assert.That(volleyBall.Name, Is.EqualTo("volley-ball"));

    // Now build a query to get all players that play volley ball, using
    // the volley ball object
    query = odb.Query<Player>();
    query.Descend("FavoriteSport").Constrain(volleyBall).Identity();

    var players = query.Execute<Player>();

    Console.WriteLine("\nStep 5 (Soda): Players of Voller-ball");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players, Has.Count.EqualTo(4));
}
```

Step 5 (LINQ to NDatabase): Example of query using object references

© 2014 Jacek Spolnik

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    // retrieve the volley ball sport object
    var volleyBall = (from sport in odb.AsQueryable<Sport>()
                      where sport.Name == "volley-ball"
                      select sport).First();

    Assert.That(volleyBall.Name, Is.EqualTo("volley-ball"));

    // Now build a query to get all players that play volley ball, using
    // the volley ball object
    var players = from player in odb.AsQueryable<Player>()
                  where player.FavoriteSport.Equals(volleyBall)
                  select player;

    Console.WriteLine("\nStep 5 (Linq): Players of Voller-ball");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players.Count(), Is.EqualTo(4));
}
```

Step 6 (SODA): Example of query using the OR operator

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var query =
        odb.Query<Player>();

    (query.Descend("FavoriteSport._name").Constrain("volley-ball").Equal())
   .Or(query.Descend("FavoriteSport._name").Constrain("%nnis").Like());

    var players = query.Execute<Player>();

    Console.WriteLine("\nStep 6 (Soda): Volley-ball and Tennis Players");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players, Has.Count.EqualTo(5));
}
```

Step 6 (LINQ to NDatabase): Example of query using the OR operator

© 2014 Jacek Spolnik

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var players = from player in odb.AsQueryable<Player>()
                  where
                      player.FavoriteSport.Name.Equals("volley-ball") ||
                      player.FavoriteSport.Name.EndsWith("nnis")
                  select player;

    Console.WriteLine("\nStep 6 (Linq): Volley-ball and Tennis Players");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players.Count(), Is.EqualTo(5));
}
```

Step 7 (SODA): Example of query using NOT and like

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var query = odb.Query<Player>();
    query.Descend("FavoriteSport._name").Constrain("volley-ball")
         .Equal().Not();

    var players = query.Execute<Player>();

    Console.WriteLine("\nStep 7 (Soda): Players that don't play Volley-ball");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players, Has.Count.EqualTo(1));
}
```

Step 7 (LINQ to NDatabase): Example of query using NOT and like

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var players = from player in odb.AsQueryable<Player>()
                  where !player.FavoriteSport.Name.Equals("volley-ball")
                  select player;

    Console.WriteLine("\nStep 7 (Linq): Players that don't play Volley-ball");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players.Count(), Is.EqualTo(1));
}
```

Step 8 (SODA): Example of query using StartsWith

© 2014 Jacek Spolnik

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var query = odb.Query<Player>();
    query.Descend("FavoriteSport._name").Constrain("VOLLEY").StartsWith(false);

    var players = query.Execute<Player>();

    Console.WriteLine("\nStep 8 (Soda): Players that play Volley-ball");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players, Has.Count.EqualTo(4));
}
```

Step 8 (LINQ to NDatabase): Example of query using StartsWith

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var players = from player in odb.AsQueryable<Player>()
                  where player.FavoriteSport.Name
                .StartsWith("VOLLEY", true, CultureInfo.InvariantCulture)
                  select player;

    Console.WriteLine("\nStep 8 (Linq): Players that play Volley-ball");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players.Count(), Is.EqualTo(4));
}
```

Step 9 (SODA): query with Where on List

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var query = odb.Query<Player>();
    query.Descend("Name").Constrain("gdalen").Contains();
    var players = query.Execute<Player>();

    var magdalena = players.GetFirst();

    // builds a query to get all teams where mihn plays
    query = odb.Query<Team>();
    query.Descend("Players").Constrain(magdalena).Contains();

    var teams = query.Execute<Team>();

    Console.WriteLine("\nStep 9 (Soda): Team where magdalena plays");

    foreach (var team in teams)
        Console.WriteLine("\t{0}", team);

    Assert.That(teams, Has.Count.EqualTo(1));
}
```

© 2014 Jacek Spolnik

Step 9 (LINQ to NDatabase): query with Where on List

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var players = from player in odb.AsQueryable<Player>()
                  where player.Name.Contains("gdalen")
                  select player;

    var magdalena = players.First();

    var teams = from team in odb.AsQueryable<Team>()
                where team.Players.Contains(magdalena)
                select team;

    Console.WriteLine("\nStep 9 (Linq): Team where magdalena plays");

    foreach (var team in teams)
        Console.WriteLine("\t{0}", team);

    Assert.That(teams.Count(), Is.EqualTo(1));
}
```

Step 10 (SODA): Ordering the query result via query

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var query = odb.Query<Player>();
    query.Descend("Name").OrderAscending();

    var players = query.Execute<Player>();

    Console.WriteLine("\nStep 10 (Soda): Players ordered by name asc");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players, Has.Count.EqualTo(5));

    query.Descend("Name").OrderDescending();
    players = query.Execute<Player>();

    Console.WriteLine("\nStep 10 (Soda): Players ordered by name desc");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players, Has.Count.EqualTo(5));
}
```

Step 10 (LINQ to NDatabase): Ordering the query result via query

© 2014 Jacek Spolnik

```
using (var odb = OdbFactory.Open(TutorialDb5MinName))
{
    var players = from player in odb.AsQueryable<Player>()
                  orderby player.Name ascending
                  select player;

    Console.WriteLine("\nStep 10 (Linq): Players ordered by name asc");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players.Count(), Is.EqualTo(5));

    players = from player in odb.AsQueryable<Player>()
              orderby player.Name descending
              select player;

    Console.WriteLine("\nStep 10 (Linq): Players ordered by name desc");

    foreach (var player in players)
        Console.WriteLine("\t{0}", player);

    Assert.That(players.Count(), Is.EqualTo(5));
}
```

Output from the tutorial

```
Step 3 : Players with name julia
   julia

Step 4 : Players of Voller-ball
   julia
   magdalena
   jacek
   michal

Step 5: Players of Voller-ball
   julia
   magdalena
   jacek
   michal

Step 6 : Volley-ball and Tennis Players
   julia
   magdalena
   jacek
   michal
   André Agassi

Step 7 : Players that don't play Volley-ball
   André Agassi

Step 8 bis: Players that play Volley-ball
   julia
   magdalena
   jacek
   michal

Step 9: Team where magdalena plays
   Team Krakow
   julia
   magdalena


Step 10: Players ordered by name asc
   André Agassi
   jacek
   julia
   magdalena
   michal

Step 10: Players ordered by name desc
   michal
   magdalena
   julia
   jacek
   André Agassi
```

3 Basic Configuration

Usually, there is no need for changing default settings of NDatabase, only logging could be the exception of that.

To set custom settings, you have to use OdbConfiguraiton static class. Take a look below on all possibilities:

© 2014 Jacek Spolnik

- Logging Configuration
- Indexes Configuration

3.1 Logging Configuration

NDatabase logging is used massively, if you want to benefit from that you can inject your logger implementation.

To enable logging in NDatabase, invoke:

```
OdbConfiguration.EnableLogging();
```

This method is turning on logging mechanism, but still you need to inject implementation of ILogger interface:

```
namespace NDatabase
{
    /// <summary>
    /// Base interface for creacting custom logger
    /// </summary>
    public interface ILogger
    {
        /// <summary>
        /// Log message with warn level
        /// </summary>
        /// <param name="message">Mssage to log</param>
        void Warning(string message);

        /// <summary>
        /// Log message with debug level
        /// </summary>
        /// <param name="message">Mssage to log</param>
        void Debug(string message);

        /// <summary>
        /// Log message with info level
        /// </summary>
        /// <param name="message">Mssage to log</param>
        void Info(string message);

        /// <summary>
        /// Log message with error level
        /// </summary>
        /// <param name="message">Mssage to log</param>
        void Error(string message);
    }
}
```

You can disable logging any time by invoking:

```
OdbConfiguration.DisableLogging();
```

and you can check if logging is already enabled:

```
var isLoggingEnabled = OdbConfiguraiton.IsLoggingEnabled();
```

© 2014 Jacek Spolnik

The last method is to register your own logger implementation:

```
var customLogger = new YourCustomLogger();
OdbConfiguration.RegisterLogger(customLogger);
```

Similar as `EnableConsoleLogger()` method, this method is turning on the logging.

For more details about logging, to see how implement log4net logger, check the logging page.

3.2 Indexes Configuration

BTree Validation

BTree indexes from performance reasons have turned off BTree validation on default. To enable this validation use:

```
OdbConfiguration.EnableBTreeValidation();
```

To check if BTree validation is enabled, invoke method:

```
var isBTreeValidationEnabled = OdbConfiguration.IsBTreeValidationEnabled();
```

And finally to disable that, use:

```
OdbConfiguration.DisableBTreeValidation();
```

BTree index degree

The default BTree indexes degree is set to **20**. To change that use below method:

```
var newTreeDegree = 30;
OdbConfiguration.SetIndexBTreeDegree(newTreeDegree);
```

To check actual value of index degree, use:

```
var degree = OdbConfiguration.GetIndexBTreeDegree();
```

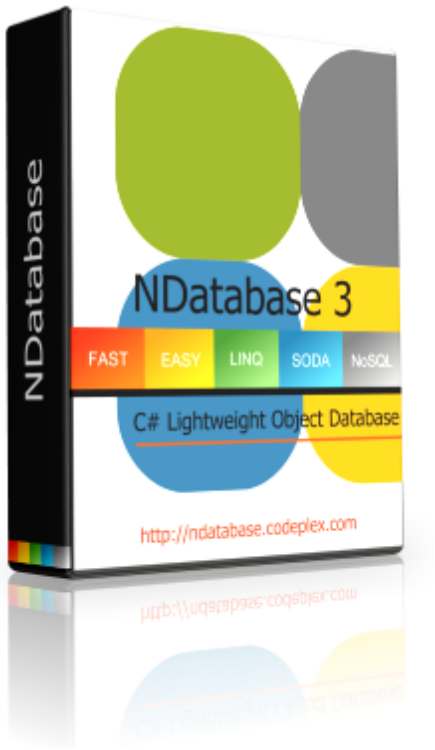
To gain more knowledge about indexes, check indexes page.

4 Features

- One single database file to store all data:
  - Meta-model
  - Objects
  - Indexes
- Handles circular references, private members (even marked as readonly)
- Safe access from many threads and from many processes
- NoSQL, Linq support and SODA queries
- Values Queries, supporting metrics like Max, Min, Avg, Count and others
- Compatible with LinqPad
- Transactions Support (ACID)
- Store any object, no need for Serializable attribute, OID, or anything else

© 2014 Jacek Spolnik

- Use as In-Memory Database
- Portability - identify types only by namespace and class name
- Zero Configuration, just reference NDatabase3.dll
- BTree Indexes
  - Unique indexes
  - Non Unique indexes
- Triggers
  - Select trigger (after)
  - Insert trigger (before, after)
  - Delete trigger (before, after)
  - Update trigger (before, after)
- Refactoring
  - Adding a new field (via API)
  - Removing a field (via API)
  - Rename a class (via API)
  - Rename a field (via API)
- Logging mechanism extensible with custom loggers
  - Sample loggers: log4net, Console
- NonPersistent attribute
- CascadeDelete attribute - allowing on cascade delete
- OID attribute - allows on mapping internal OID to class defined field
- Sample Northwind database
- Handle Assembly Version Changes
- Dynamic Schema Evolution
- Source Code Available
- Supported Platforms
  - Microsoft .net 3.5, 4.0, 4.5
  - Silverlight 4, Silverlight 5 (see limitations)
  - Windows Phone 7.1, Windows Phone 7.5 (see limitations)
  - Mono (4.0)
  - NuGet



4.1 SODA Query

The SODA query API is one of NDatabase low level querying API, allowing direct access to nodes of query graphs. Since SODA uses strings to identify fields, it is neither perfectly type safe nor compile-time checked and it also is quite verbose to write.

For most applications LINQ will be the better querying interface. However there can be applications where dynamic generation of queries is required, or you want to have an access to private or internal fields.

SODA and Values Query are also an underlying NDatabase querying mechanism. LINQ query syntax is translated to SODA under the hood.

Understanding SODA will provide you with a better understanding of NDatabase in the whole and will help to write queries and applications with better performance.

See more:
- Building SODA Queries

© 2014 Jacek Spolnik

- SODA Query Graph
- SODA Query API
- Execute SODA Query

Origin of the soda

S.O.D.A. is a an object API to communicate with databases.The current specification is focused on queries.
Goals are: type safety, object reuse, minimum string use, programming language independance.

Home page: http://sodaquery.sourceforge.net/
Sourceforge page: http://sourceforge.net/projects/sodaquery/

The original SODA queries definition is slightly changed in NDatabase.

4.1.1 Building SODA Queries

During the code samples, we will use classes predefined in Basic Sample.

A new Query object is created through the Query method of the IOdb and we can add IConstraint instances to it. To find all Warrior instances, we need to invoke generic Query method with type as Warrior class.

```
public void GetAllWarriors()
{
    IOdb odb = OdbFactory.Open(DbName);
    try
    {
        IQuery query = odb.Query<Warrior>();
        IObjectSet<Warrior> result = query.Execute<Warrior>();
        PrintResult(result);
    }
    finally
    {
        odb.Close();
    }
}
```

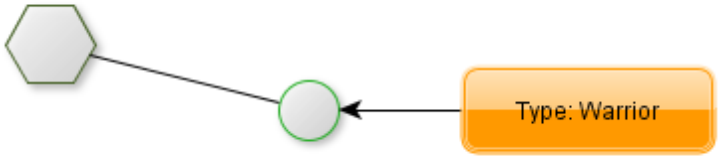
Additionally, for the case when you are interested only in retrieving all instances of given type, you can use simplified form:

```
IObjectSet<Warrior> result = query.QueryAndExecute<Warrior>();
```

Basically, we are using meta description of the objects we'd like to hunt down: a query graph made up of query nodes and constraints. A query node is a placeholder for a candidate object, a constraint decides whether to add or exclude candidates from the result. In above simple case, we constrain the type during creation of the query.

Our first simple graph looks like this:

© 2014 Jacek Spolnik

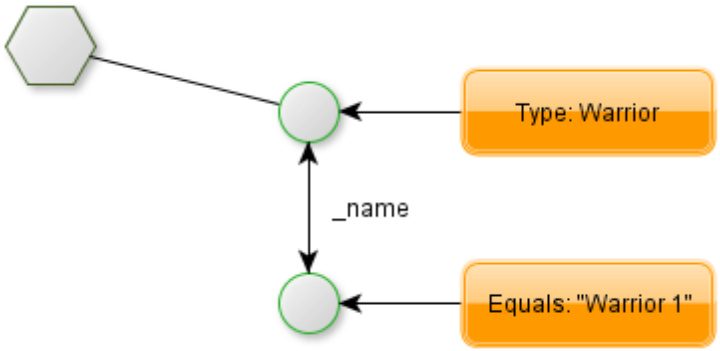


We're just asking any candidate object (here: any object in the database) to be of type Warrior to aggregate our result.

To get a warrior by name, we have to further constrain the candidate warriors by descending to their name field and constraining this with the respective candidate String.

```
public void GetWarriorByName()
{
    IOdb odb = OdbFactory.Open(DbName);
    try
    {
        IQuery query = odb.Query<Warrior>();
        query.Descend("_name").Constrain("Warrior 1").Equal();
        IObjectSet<Warrior> result = query.Execute<Warrior>();
        PrintResult(result);
    }
    finally
    {
        odb.Close();
    }
}
```

What does 'descend' mean here? Well, we can attach constraints to child members of our candidates.

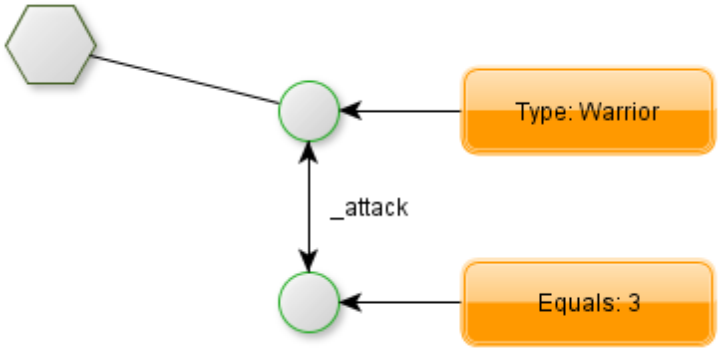


So a candidate needs to be of type Warrior and have a member named '_name' that is equal to the given String to be accepted for the result.

Finding a warrior by exact attack value is analogous:

© 2014 Jacek Spolnik

```
public void GetWarriorByExactAttackValue()
{
    IOdb odb = OdbFactory.Open(DbName);
    try
    {
        IQuery query = odb.Query<Warrior>();
        query.Descend("_attack").Constrain(3).Equal();
        IObjectSet<Warrior> result = query.Execute<Warrior>();
        PrintResult(result);
    }
    finally
    {
        odb.Close();
    }
}
```



Inheritance

In the case of inherited classes of interfaces:
- querying against parent class or interface will include results for all subclasses/implementations

4.1.2 SODA Query Graph

SODA allows to create a query graph of any complexity by joining field object constraints. SODA usage is very generic and can be applied to any objects and conditions. The following 5 steps can be used (all steps are optional and can be repeated logically) in the following order:

1. Create a root of a query object

```
IQuery query = odb.Query<Foo>();
```

2. Navigate from any query node to any subordinate node

```
IQuery barNode = query.Descend("bar");
```

3. Add constraints to any node

```
IConstraint barConstraint = barNode.Constrain(5);
```

4. Set the evaluation mode of a node

© 2014 Jacek Spolnik

```
barConstraint().Greater();
```

5. Execute query

```
IObjectSet<Foo> result = query.Execute<Foo>();
```

Additionally, if you are doing only 1 and 5 steps (so you want to get all Foo instances from database), you can use simplified syntax:

```
IObjectSet<Foo> result = odb.QueryAndExecute<Foo>();
```

The API is very powerful with a small number of method calls.
The "backward" order to add constraints first and to specify the evaluation mode as a second step allows plugging complex objects into a query.

4.1.3 SODA Query API

During the code samples, we will use classes predefined in Basic Sample.

There are occasions when we don't want to query for exact field values, but rather for value ranges, objects not containing given member values, etc. This functionality is provided by the Constraint API.

Not

First, let's negate a query to find all warriors who are not "Warrior 1":

```
public void GetWarriorByNegation()
{
    IOdb odb = OdbFactory.Open(DbName);
    try
    {
        IQuery query = odb.Query<Warrior>();
        query.Descend("_name").Constrain("Warrior 1").Equal().Not();
        IObjectSet<Warrior> result = query.Execute<Warrior>();
        PrintResult(result);
    }
    finally
    {
        odb.Close();
    }
}
```

And

Where there is negation, the other boolean operators can't be too far:

© 2014 Jacek Spolnik

```
public void GetWarriorByConjunction()
{
    IOdb odb = OdbFactory.Open(DbName);
    try
    {
        IQuery query = odb.Query<Warrior>();
        var nameConstraint = query.Descend("_name").Constrain("Warrior 1").Equal();
        query.Descend("_attack").Constrain(3).And(nameConstraint);
        IObjectSet<Warrior> result = query.Execute<Warrior>();
        PrintResult(result);
    }
    finally
    {
        odb.Close();
    }
}
```

Or

```
public void GetWarriorByDisjunction()
{
    IOdb odb = OdbFactory.Open(DbName);
    try
    {
        IQuery query = odb.Query<Warrior>();
        var nameConstraint = query.Descend("_name").Constrain("Warrior 1").Equal();
        query.Descend("_attack").Constrain(3).Or(nameConstraint);
        IObjectSet<Warrior> result = query.Execute<Warrior>();
        PrintResult(result);
    }
    finally
    {
        odb.Close();
    }
}
```

Greater, Smaller, Equal <>=

We can also constrain to a comparison with a given value. Return warriors with more than 2 attack points:

```
public static void GetWarriorsByComparison()
{
    IOdb odb = OdbFactory.Open(DbName);
    try
    {
        IQuery query = odb.Query<Warrior>();
        query.Descend("_attack").Constrain(2).Greater();
        IObjectSet<Warrior> result = query.Execute<Warrior>();
        PrintResult(result);
    }
    finally
    {
        odb.Close();
    }
}
```

© 2014 Jacek Spolnik

String Comparisons - Like and Contains

This is an equivalent to SQL "like" operator.

```
public void GetWarriorsByLikeOrContains()
{
    IOdb odb = OdbFactory.Open(DbName);
    try
    {
        IQuery query2 = odb.Query<Warrior>();

        query2.Descend("_name").Constrain("rior").Contains();
        IObjectSet<Warrior> result1 = query2.Execute<Warrior>();
        PrintResult(result1);


        IQuery query3 = odb.Query<Warrior>();

        query3.Descend("_name").Constrain("rior").Like();
        IObjectSet<Warrior> result2 = query3.Execute<Warrior>();
        PrintResult(result2);
    }
    finally
    {
        odb.Close();
    }
}
```

String Comparisons - StartsWith, EndsWith

Compares a beginning or ending of a string:

```
public void GetWarriorsByStartsOrEndsWith()
{
    IOdb odb = OdbFactory.Open(DbName);
    try
    {
        IQuery query2 = odb.Query<Warrior>();
        query2.Descend("_name").Constrain("Warr").StartsWith(false);
        IObjectSet<Warrior> result1 = query2.Execute<Warrior>();
        PrintResult(result1);

        IQuery query3 = odb.Query<Warrior>();
        query3.Descend("_name").Constrain("rior 1").EndsWith(false);
        IObjectSet<Warrior> result2 = query3.Execute<Warrior>();
        PrintResult(result2);
    }
    finally
    {
        odb.Close();
    }
}
```

© 2014 Jacek Spolnik

4.1.4 Execute SODA Query

Normal mode

- Execute<T>() - allows on executing any SODA query

Memory set mode

- Execute<T>(bool inMemory) - allows on executing any SODA query and indicates if all returned data should be loaded to memory (inMemory is true) or if the data should be lazy loaded (inMemory to false)

Memory set mode limiting result size -

- IObjectSet<T> Execute<T>(bool inMemory, int startIndex, int endIndex) - allows on executing any SODA query and indicates if all returned data should be loaded to memory (inMemory is true) or if the data should be lazy loaded (inMemory to false). Additionally we could limit result by setting start and end index of it.

4.2 Values Query API

The Values Query API is one of NDatabase low level querying API extending query with additional calculation metrics, allowing on calculating them directly on nodes of query graphs. It is fully compatible with SODA Query API which allows on filtering the result set on which we want to calculate the metric.

Available operations (see IValuesQuery):

- Sum - calculate the sum of the specified field name.
- Count - counts the objects that matches the specified values query.
- Avg - calculate average of the specified field name.
- Max - calculate max value of the specified field name.
- Min - calculate min value of the specified field name.
- Field - queries for field values.
- Sublist - calculate sublist of the specified attribute name (collection or string).
- Size - calculate size of the specified attribute name (collection or string).
- GroupBy - groups the result by the specified field list.

Additionally, we could use method SetReturnInstance to decide, if we want to work with instances or just with object representations (lightweight meta representation of objects - IObjectRepresentation).

Working with Values Query API

To create the values query, you need to use IOdb instance in the same way as normal queries in NDatabase:

```
using (var odb = OdbFactory.Open(DbName))
 {
     var valuesQuery = odb.ValuesQuery<SomeClass>();
     // more code ...
 }
```

There is minor difference regarding executing values query in comparison to normal query: you **cannot** use generic execute methods, because they will cause unsupported operation exception (OdbRuntimeException). For values query, there is much simple Execute method which needs to be invoked to take the IValues result.

© 2014 Jacek Spolnik

The returned set implements mentioned IValues interface:

```
/// <summary>
///   The main interface of all Object Values query results of NDatabase ODB
/// </summary>
public interface IValues : IObjectSet<IObjectValues>
{
   /// <summary>
   /// Get next values set
   /// </summary>
   /// <returns>Next values</returns>
   IObjectValues NextValues();
}
```

This interface could be used in the same as normal result from NDatabase queries, or we could wrap results in IObjectValues implementation by invoking NextValues method:

```
/// <summary>
///   Interface that will be implemented to hold a row of a result of an Object Values Query
/// </summary>
public interface IObjectValues
{
    /// <summary>
    /// Get result by alias
    /// </summary>
    /// <param name="alias">Alias for result</param>
    /// <returns>Object result</returns>
    object GetByAlias(string alias);

    /// <summary>
    /// Get result by index
    /// </summary>
    /// <param name="index">Index for result</param>
    /// <returns>Object result</returns>
    object GetByIndex(int index);

    /// <summary>
    /// Get values result
    /// </summary>
    /// <returns>Array of objects as values</returns>
    object[] GetValues();
}
```

It allows us on just take the values as objects array, get them by the index or, what's more important, take the value by alias. Aliasing allows us on easier manipulation on the results, we could calculate many metrics and give them aliases which will make easier to query them from result set.

All operations are supporting aliasing, only GroupBy isn't, which make sense because of its nature. It should be used with other metric, for which we want to group the results.

Examples

Class used through examples:

© 2014 Jacek Spolnik

```
public class Result
{
    private readonly string _name;
    private readonly int _value;
    private readonly string _category;

    public Result(string name, int value, string category)
    {
        _name = name;
        _value = value;
        _category = category;
    }

    public string GetCategory()
    {
        return _category;
    }

    public string GetName()
    {
        return _name;
    }

    public int GetValue()
    {
        return _value;
    }
}
```

Below code was used to generate sample data set, for calculating below metrics:

```
using (var odb = OdbFactory.Open(DbName))
{
   for (var i = 1; i <= Limit; i++)
      odb.Store(new Result("Result" + i, i, (i % 2 == 0) ? "EVEN" : "ODD"));
}
```

- **Sum**

```
IValues values;

using (var odb = OdbFactory.Open(DbName))
{
    values = odb.ValuesQuery<Result>().Sum("_value", "sum").Execute();
}

var objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("sum"), Is.EqualTo(55m));
```

- **Min**

© 2014 Jacek Spolnik

```
IValues values;

using (var odb = OdbFactory.Open(DbName))
{
    values = odb.ValuesQuery<Result>().Min("_value", "min").Execute();
}

var objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("min"), Is.EqualTo(1m));
```

- **Max**

```
IValues values;

using (var odb = OdbFactory.Open(DbName))
{
    values = odb.ValuesQuery<Result>().Max("_value", "max").Execute();
}

var objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("max"), Is.EqualTo(10m));
```

- **Avg**

```
IValues values;

using (var odb = OdbFactory.Open(DbName))
{
    values = odb.ValuesQuery<Result>().Avg("_value", "avg").Execute();
}

var objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("avg"), Is.EqualTo(5.5m));
```

- **Count**

```
IValues values;

using (var odb = OdbFactory.Open(DbName))
{
    values = odb.ValuesQuery<Result>().Count("count").Execute();
}

var objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("count"), Is.EqualTo(10m));
```

- **Field**

© 2014 Jacek Spolnik

```
IValues values;

using (var odb = OdbFactory.Open(DbName))
{
    values = odb.ValuesQuery<Result>().Field("_value", "field").Execute();
}

var objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(1));

objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(2));

objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(3));

objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(4));

objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(5));

objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(6));

objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(7));

objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(8));

objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(9));

objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(10));

objectValues = values.NextValues();
Assert.That(objectValues, Is.Null);
```

- **Sublist (string)**

```
IValues values;

using (var odb = OdbFactory.Open(DbName))
{
    var valuesQuery = odb.ValuesQuery<Result>();
    values = valuesQuery.Sublist("_name", "sublist", 4, 3, true).Execute();
}

var objectValues = values.NextValues();
var firstNameSublist = (IList)objectValues.GetByAlias("sublist");

var stringBuilder = new StringBuilder();

foreach (var value in firstNameSublist)
    stringBuilder.Append(value);

Assert.That(stringBuilder.ToString(), Is.EqualTo("lt1"));
```

© 2014 Jacek Spolnik

- **Size (string)**

```
IValues values;

using (var odb = OdbFactory.Open(DbName))
{
    values = odb.ValuesQuery<Result>().Size("_name", "size").Execute();
}

var objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("size"), Is.EqualTo(7));
```

- **GroupBy**

```
IValues values;

using (var odb = OdbFactory.Open(DbName))
{
    values = odb.ValuesQuery<Result>().Sum("_value", "sum_group").GroupBy("_category").Execute();
}

//ODD
var objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("sum_group"), Is.EqualTo(25m));

//EVEN
objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("sum_group"), Is.EqualTo(30m));
```

- **Field and SODA query**

```
IValues values;

using (var odb = OdbFactory.Open(DbName))
{
    var valuesQuery = odb.ValuesQuery<Result>();
    valuesQuery.Descend("_name").Constrain("Result4");
    values = valuesQuery.Field("_value", "field").Execute();
}

var objectValues = values.NextValues();
Assert.That(objectValues.GetByAlias("field"), Is.EqualTo(4));

objectValues = values.NextValues();
Assert.That(objectValues, Is.Null);
```

4.3 LINQ to NDatabase

Enter topic text here.

© 2014 Jacek Spolnik

4.4 LinqPad Support

NDatabase is supporting LinqPad, as the viewer for data.
You can use Linq to Objects, or Linq to NDatabase to query database data and then view them in LinqPad.

Here you can find:
- how to install LinqPad for NDatabase
- how to create own typed data context

Additionally, you could be interested in checking Northwind sample with LinqPad as the part of it.

© 2014 Jacek Spolnik

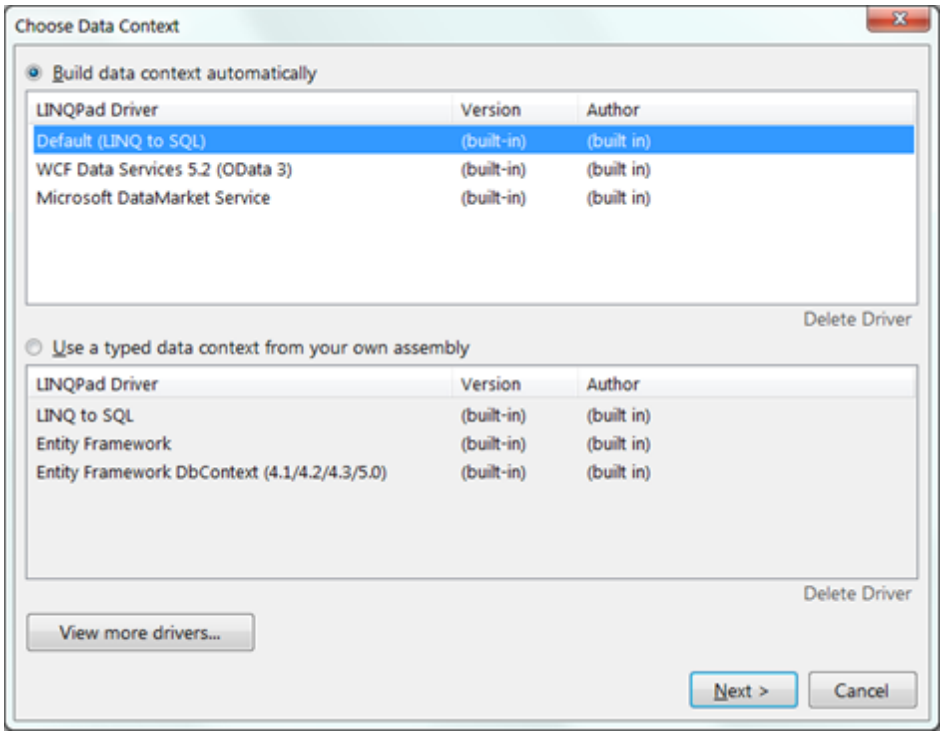
4.4.1 LinqPad for NDatabase - installation

Prerequisites

- Download free LinqPad version from project home page.
- Download ndatabaselinqpaddriver_3.lpx LinqPad driver.

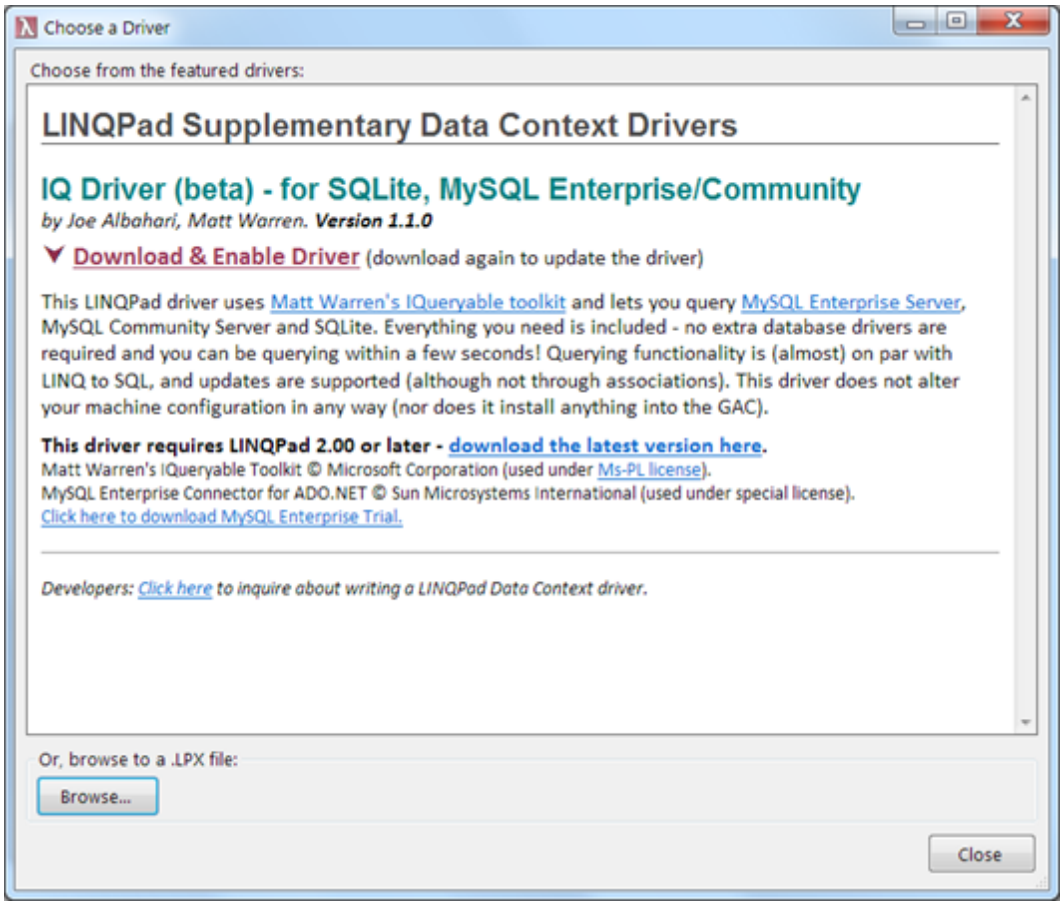
Install static driver

Open LinqPad, then click “Add Connection”, the following dialog appears:



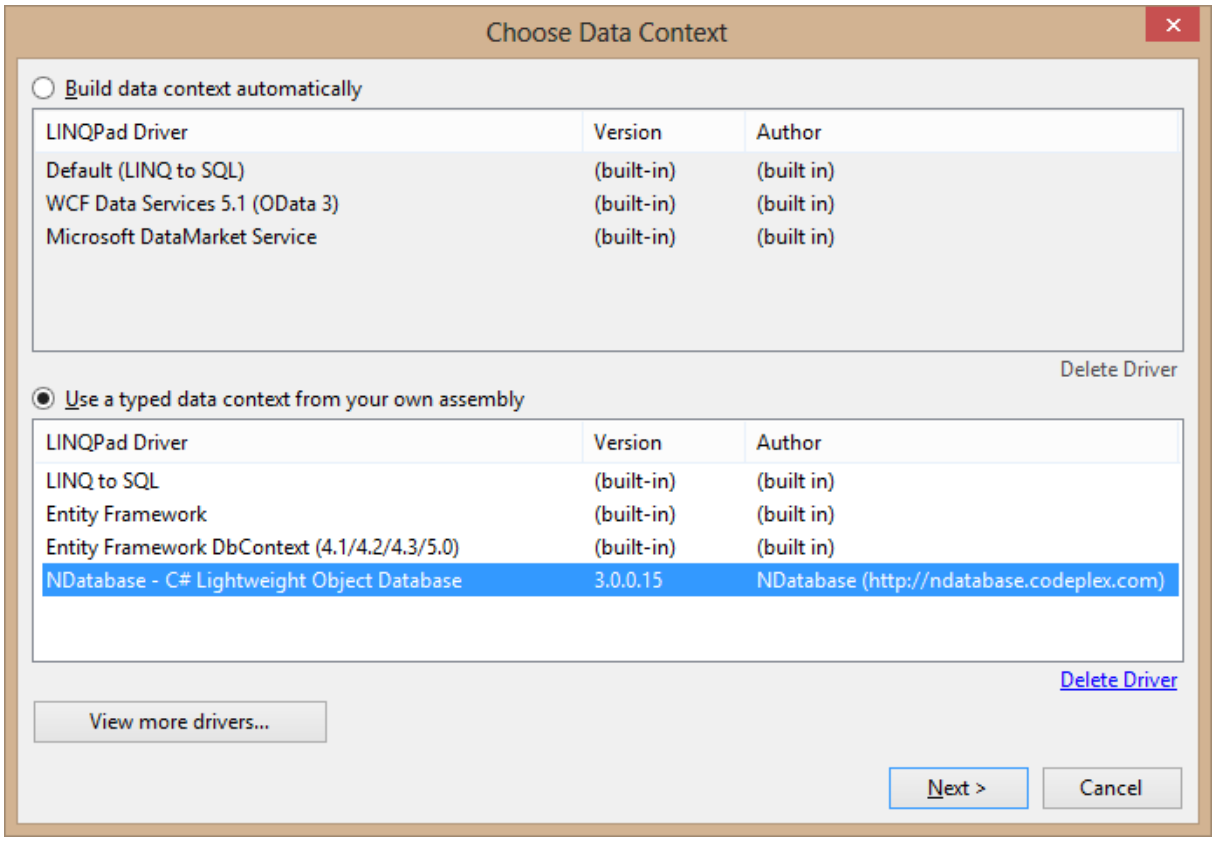
Clicks “View More Drivers”, the following dialog appears:

© 2014 Jacek Spolnik



Click the “Browse" button and import a **ndatabaselinqpaddriver_3.lpx** file. Once clicked, the driver will become visible in the previous dialog, and you can create new NDatabase connection.

© 2014 Jacek Spolnik



4.4.2 LinqPad - Creating Typed Data Context

A typed data context is a class with properties/fields/methods that the user can query. A classic example is a typed LINQ to SQL DataContext (or a typed ObjectContext in Entity Framework):

```
public class TypedDataContext : DataContext
{
      public IQueryable<Customer> Customers
      { get { return this.GetTable<Customer>(); } }

      public IQueryable<Order> Orders
      { get { return this.GetTable<Orders>();   } }
}
```

A typed data context does not need base class. The following is perfectly valid:

```
public class TypedDataContext
{
      public IEnumerable<string> CustomerNames
      public int[] Numbers;
      public void DoSomething() { … }
}
```

© 2014 Jacek Spolnik

4.5 Indexes

How to use indexes

To use index, firstly you need to invoke method IndexManagerFor<T> on IOdb instance, with T as the class for which field you want to add index.

Method returns the instance of IIndexManager interface, which allows us on:
- adding new named unique index on defined fields (`AddUniqueIndexOn`)
- adding new named index on defined fields (`AddIndexOn`)
- checking if named index already exists (`ExistIndex`)
- rebuilding named index (`RebuildIndex`)
- deleting named index (`DeleteIndex`)

Sample code

```
using (var odb = OdbFactory.Open("index1.ndb"))
{
    odb.IndexManagerFor<Player>().AddUniqueIndexOn("nameIndex", "Name");
    odb.IndexManagerFor<Game>().AddIndexOn("nameIndex", "Result");

    // more code ...
}
```

Configuration

To check configuration possibilities for indexes, please check indexes configuration page.

Implementation

Indexes are implemented using B-Tree structure.
- The namespace NDatabase.BTree contains a generic B-Tree implementation.
- Index definitions are stored in NDatabase as normal objects of class `NDatabase.Meta.ClassInfoIndex` (this internal class is marked as a system class).
- Index content is stored in NDatabase as `NDatabase.BTree.IBTree` and `NDatabase.BTree.IBTreeNode` (internal classes).

4.6 Transactions

NDatabase uses transactions to guarantee database integrity (ACID properties).

When an NDatabase database is opened, a session is automatically created. The session contains a special object representing transaction. The transaction instance manage all the data and behavior of the current transaction.

When executing an operation (insert, update, delete) that modifies the database, some parts of the operation can be directly written to the database file (buffers for IO performance) other can not.

Example

© 2014 Jacek Spolnik

When inserting a new object, the object can be written directly to the database but references to this object can not. In this case, references are written in the transaction (as a list of write actions).

So basically, NDatabase is being optimistic and write almost everything to the database. Only pointers and data from update and delete are written in the Transaction. The writing in Transaction begins as soon as the implicit transaction is created by NDatabase. On commit, the transaction content is applied to main database file.

The Transaction keep a list of write actions (that contain data that can not be written directly) in memory. On Commit, if all write actions are on memory, NDatabase simply apply them to the database file.

If not, NDatabase first loads them from the Transaction and then apply them to the database file. After commit, the transaction is deleted.

Commit Process

The database can be divided into 2 different zones:
- Committed Zone - the committed zone contains all committed work.
- Uncommitted (Unconnected) zone - uncommitted zone contains all the work that has not been committed yet,work of open transactions. All new objects and modifications must be created in the uncommitted zone.

Uncommitted zone can be divided in 2 sub-one:
- Unconnected Zone - new objects zone.
- Connected Zone - modifications zone.

The Unconnected Zone contains all the new objects without any link (pointers) from the committed zone. All new objects are actually written in the database file and will stay in the database in case of rollback but they will be unreachable so it will not bring inconsistency!

The Connected Zone contains write actions (that represents object updates and deletes) that will be applied to the committed zone when committed.

When Committing

- Uncommitted Transaction Zone - all first new objects must be connected to last committed objects (previous object OID and next object OID). This must be done for all new objects types.
- Connected TransactionZone - write actions must be applied to the committed zone.

The commit operation must use a Storage Engine Scope Lock.

Example

Firstly, we have to prepare to objects, which we will use in the sample. Below you can see the class of which the objects we will create.

© 2014 Jacek Spolnik

```
public class SimpleClass
{
   public string Name { get; set; }
   public int Value { get; set; }
}
```

An the mentioned objects:

```
var a = new SimpleClass();
a.Name = "abc";
a.Value = 3;

var b = new SimpleClass();
b.Name = "def";
b.Value = 6;
```

Now, we will store both objects. After storing **a** object we will commit changes, after storing **b** object we will rollback changes. That will allow us on saving part of changes, and on reverting the others.

```
using (var odb = OdbFactory.Open(DbName))
{
   odb.Store(a);
   odb.Commit();
   odb.Store(b);
   odb.Rollback();
}
```

As we can see in below assertions, only one object (a) is stored, and all values of it are equal to what we stored.

```
using (var odb = OdbFactory.Open(DbName))
{
   var items = odb.QueryAndExecute<SimpleClass>();
   Assert.That(items.Count, Is.EqualTo(1));

   var item = items.GetFirst();
   Assert.That(item.Name, Is.EqualTo("abc"));
   Assert.That(item.Value, Is.EqualTo(3));
}
```

Here, we are changing both properties of the stored object, but only first update will be stored because we commited it. Change of the value was reverted, as you can see in below code:

```
using (var odb = OdbFactory.Open(DbName))
{
   var item = odb.QueryAndExecute<SimpleClass>().GetFirst();
   item.Name = "ghi";
   odb.Store(item);
   odb.Commit();
   item.Value = 9;
   odb.Store(item);
   odb.Rollback();
}
```

Final results:

© 2014 Jacek Spolnik

```
using (var odb = OdbFactory.Open(DbName))
{
   var item = odb.QueryAndExecute<SimpleClass>().GetFirst();
   Assert.That(item.Name, Is.EqualTo("ghi"));
   Assert.That(item.Value, Is.EqualTo(3));
}
```

4.7 In Memory Database

You can use NDatabase as in memory database. All data will be stored directly to memory, that's included all transactions too.

Below you can find the sample, how to use in memory mode:

```
using (var inMemory = OdbFactory.OpenInMemory())
{
    var inMemoryTestClass = new
        InMemoryTestClass {Name = "Test", Size = 1};

    inMemory.Store(inMemoryTestClass);

    var query = inMemory.Query<InMemoryTestClass>();
    query.Descend("Name").Constrain("Test").Equal();

    var items = query.Execute<InMemoryTestClass>();

    var memoryTestClass = items.FirstOrDefault();

   // Further processing
}
```

In above code, we used InMemoryTestClass:

```
public class InMemoryTestClass
{
    public string Name { get; set; }
    public int Size { get; set; }
}
```

4.8 Triggers

To add a trigger, the first step is to build a class that implements one of the trigger interface.

- Trigger, base abstract class,
- Insert Trigger,
- Update Trigger,
- Select Trigger,
- Delete Trigger.

Update trigger contains IObjectRepresentation item which is used to give the user an instance of an object representation. The Object Representation encapsulates the NonNativeObjectInfo which is the internal object representation.

Insert Trigger Example

© 2014 Jacek Spolnik

```
public class MyTrigger : InsertTrigger
{
    public override bool BeforeInsert(object obj)
    {
        Console.WriteLine("before inserting {0}", obj);
        ((Mage) obj).Attack += 1.1;
        Console.WriteLine("before inserting (after change) {0}", obj);

        return true;
    }

    public override void AfterInsert(object obj, OID oid)
    {
        Console.WriteLine("after insert object with id {0}({1})", oid,
                obj.GetType().Name);
    }
}
```

You need to register the trigger:

```
var mage = new Mage("Merlin", 3.3, 3.4);
var myTrigger = new MyTrigger();

using (var odb = OdbFactory.Open("inserting_trigger.db"))
{
    odb.TriggerManagerFor<Mage>().AddInsertTrigger(myTrigger);
    odb.Store(mage);
}

using (var odb = OdbFactory.Open("inserting_trigger.db"))
{
    var merlin = odb.Query<Mage>().Execute<Mage>().GetFirst();

    // code which is using merlin Mage instance
}
```

The output from example:

```
before inserting [Mage] Name: Merlin, Attack: 3,3, Defense: 3,4
before inserting (after change) [Mage] Name: Merlin, Attack: 4,4, Defense: 3,4
after insert object with id 2(Mage)
```

4.9 Refactoring

During database lifetime, your class definitions could be change. NDatabase provides refactor manager, which is helping to solve these changes.

To get the instance of IRefactorManager, you need to use below code:

```
using (var odb = OdbFactory.Open("dbname.ndb"))
{
   var refactorManager = odb.GetRefactorManager();
   // further code ...
}
```

Refactor manager allows on:

- renaming field of stored class
- renaming stored class

© 2014 Jacek Spolnik

- adding field to stored class
- remove field from stored class

Example

User class before changes

```
public class User
{
    private int age;
    private string name;

    public string Name
    {
        get { return name; }
        set { name = value; }
    }

    public int Age
    {
        get { return age; }
        set { age = value; }
    }
}
```

Code to store class instance

```
var user = new User {Name = "Jacek", Age = 25};

using (var odb = OdbFactory.Open("Refactoring.odb"))
    odb.Store(user);
```

User class after changes

```
public class User
{
    private int _age;
    private string _name;

    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }

    public int Age
    {
        get { return _age; }
        set { _age = value; }
    }
}
```

Code to rename fields (following the change we did)

© 2014 Jacek Spolnik

```
using (var odb = OdbFactory.Open("Refactoring.odb"))
{
    var refactorManager = odb.GetRefactorManager();
    refactorManager.RenameField(typeof(User), "age", "_age");
    refactorManager.RenameField(typeof(User), "name", "_name");
}
```

Code to check if our refactoring is working well

```
using (var odb = OdbFactory.Open("Refactoring.odb"))
{
    var users = odb.GetObjects<User>();
    Assert.That(users, Has.Count.EqualTo(1));

    var first = users.GetFirst();
    Assert.That(first.Name, Is.EqualTo("Jacek"));
    Assert.That(first.Age, Is.EqualTo(25));
}
```

4.10 Logging

To enable or disable logging, add own logger - check logging configuration page.

Using internal console logger

To enable internal console logger, we just need to invoke one method. This method will add the logger into NDatabase engine and automatically it will turn on logging for NDatabase.

```
OdbConfiguration.EnableConsoleLogger();
```

Using custom logger (log4net example)

To use your logging library with NDatabase, you need to add own custom logger implementation and then register it through OdbConfiguration class. Again, that will automatically turn on logging for NDatabase.

Below you have log4net custom logger, which allows us to reuse existing log4net framework for logging purposes in NDatabase

© 2014 Jacek Spolnik

```
public class Log4NetLogger : ILogger
{
    private static readonly ILog Log = LogManager.GetLogger(typeof (ILogger));

    #region Implementation of ILogger

    public void Warning(string message)
    {
        Log.Warn(message);
    }

    public void Debug(string message)
    {
        Log.Debug(message);
    }

    public void Info(string message)
    {
        Log.Info(message);
    }

    public void Error(string message)
    {
        Log.Error(message);
    }

    public void Error(string message, Exception t)
    {
        Log.Error(message);
        Log.ErrorFormat("Error: {0}, exception: {1}", t.Message, t);
    }

    #endregion
}
```

Then we need to register this class instance:

```
OdbConfiguration.RegisterLogger(new Log4NetLogger());
```

4.11 Exceptions

NDatabase uses an own exception type to handle errors : ODBRuntimeException

The constructor of that handles an instance of IError (internal) which describes the reason of exception.

Specific NDatabase Exceptions

- BTreeException - exception particular for BTree implementation in NDatabase
- BTreeNodeValidationException - exception caused by validation error in BTrees engine
- DuplicatedKeyException - derives from BTreeException, it is used by Unique Indexes when duplicated key is inserted into index
- CorruptedDatabaseException - exception raised when database file corruption is discovered
- LinqQueryException - appears if during Linq Query execution the exception will appear

© 2014 Jacek Spolnik

4.12 Extended API

Some methods of NDatabase are grouped in a separate interface IOdbExt instead of NDatabase to keep the main interface NDatabase simple.

To get access to the extended API, just call IOdb.Ext(). Here is an example:

```
var odb = OdbFactory.Open("extb.ndb");
var f = new Function("Test Function");
var oid = odb.Store(f);
var extOid = odb.Ext().GetObjectExternalOID(f);
// code using extOid ...
odb.Close();
```

4.13 Non Persistent Attribute

The attribute was created for reasons like:

- confidential data
- temporary values not needed to be persistent
- any other custom requirement

To apply it, you just need to mark field by attribute:

```
public class User
{
    private readonly string _name;

    [NonPersistent]
    private readonly string _password;

    public User(string name, string password)
    {
        _name = name;
        _password = password;
    }

    public string Name
    {
        get { return _name; }
    }

    public string Password
    {
        get { return _password; }
    }
}
```

After storing User class instance, password field will be omitted from this process (will have default value which is in this case null).

4.14 Cascade Delete Attribute

The aim of attribute is to allow on cascade delete.

Example

© 2014 Jacek Spolnik

Class with field marked with cascade delete attribute:

```
[CascadeDelete]
private readonly Address _address1;

private readonly Address _address2;

public TestCascadeDeleteClass(Address address1, Address address2)
{
    _address1 = address1;
    _address2 = address2;
}

public Address Address1
{
    get { return _address1; }
}

public Address Address2
{
    get { return _address2; }
}
```

Working sample:

```
OdbFactory.Delete(DbName);

using (var odb = OdbFactory.Open(DbName))
{
    var shouldBeDeleted = new Address("A", City.Cracow, 1);
    var shouldNotBeDeleted = new Address("B", City.Cracow, 1);
    odb.Store(new TestCascadeDeleteClass(shouldBeDeleted, shouldNotBeDeleted));
}

using (var odb = OdbFactory.Open(DbName))
{
    var count = odb.QueryAndExecute<Address>().Count;
    Assert.That(count, Is.EqualTo(2));
}

using (var odb = OdbFactory.Open(DbName))
{
    var first = odb.QueryAndExecute<TestCascadeDeleteClass>().GetFirst();
    odb.Delete(first);
}

using (var odb = OdbFactory.Open(DbName))
{
    var count = odb.QueryAndExecute<TestCascadeDeleteClass>().Count;
    Assert.That(count, Is.EqualTo(0));

    count = odb.QueryAndExecute<Address>().Count;
    Assert.That(count, Is.EqualTo(1));

    var first = odb.QueryAndExecute<Address>().GetFirst();
    Assert.That(first.Street, Is.EqualTo("B"));
}
```

In this example we could see that one address is deleted while we deleted only the parent (the one marked with cascade delete attribute), and the second one still exists in database.

© 2014 Jacek Spolnik

4.15 OID Mapping Attribute

The aim of attribute is to allow on mapping internal Object Id (OID) to exiting field of class.

We could apply OID attribute to fields of type:
- OID - we map whole OID object to our class field
- long - we map underlying long number assigned to OIS instance

Example

Class with long field marked with OID attribute:

```
public class TestOidClassLong
{
    [OID]
    private readonly long _oid;

    public TestOidClassLong(string value)
    {
        Value = value;
        _oid = -1;
    }

    public string Value { get; set; }

    public long OID
    {
        get { return _oid; }
    }
}
```

Usage (we have long number assigned to our field):

```
OdbFactory.Delete(DbName);
const string value = "value";

using (var odb = OdbFactory.Open(DbName))
{
    odb.Store(new TestOidClassLong(value));
}

using (var odb = OdbFactory.Open(DbName))
{
    var first = odb.QueryAndExecute<TestOidClassLong>().GetFirst();

    Assert.That(first.OID, Is.GreaterThan(0L));
    Assert.That(first.Value, Is.EqualTo(value));
}
```

Class with OID field marked with OID attribute:

© 2014 Jacek Spolnik

```
public class TestOidClassOID
{
    [OID]
    private readonly Api.OID _oid;

    public TestOidClassOID(string value)
    {
        Value = value;
        _oid = null;
    }

    public string Value { get; set; }

    public Api.OID OID
    {
        get { return _oid; }
    }
}
```

Usage (we have OID instance assigned to our field):

```
OdbFactory.Delete(DbName);
const string value = "value";

using (var odb = OdbFactory.Open(DbName))
{
    odb.Store(new TestOidClassOID(value));
}

using (var odb = OdbFactory.Open(DbName))
{
    var first = odb.QueryAndExecute<TestOidClassOID>().GetFirst();

    Assert.That(first.OID, Is.Not.Null);
    Assert.That(first.OID.ObjectId, Is.GreaterThan(0));
    Assert.That(first.Value, Is.EqualTo(value));
}
```

Class with other type of field marked with OID attribute (in this case string):

```
public class TestOidClassString
{
    [OID]
    private readonly string _oid;

    public TestOidClassString(string value)
    {
        Value = value;
        _oid = null;
    }

    public string Value { get; set; }

    public string OID
    {
        get { return _oid; }
    }
}
```

Usage (when we assign OID attribute to field which is not of type long nor OID, the value of this field is

© 2014 Jacek Spolnik

not changed):

```
OdbFactory.Delete(DbName);
const string value = "value";

using (var odb = OdbFactory.Open(DbName))
{
    odb.Store(new TestOidClassString(value));
}

using (var odb = OdbFactory.Open(DbName))
{
    var first = odb.QueryAndExecute<TestOidClassString>().GetFirst();

    Assert.That(first.OID, Is.Null);
    Assert.That(first.Value, Is.EqualTo(value));
}
```

4.16 Portability

NDatabase could work in two modes regarding Type Resolution: normal mode and less restricted mode. The change between this two options is automatic.

Types Resolution Normal Mode

During the normal mode, all types are identifies with assembly qualified names. That's mean, every type is identified by namespace, class name and assembly name. It is the natural way of resolving types in CLR, and that is the default mode in NDatabase - NDatabase always store assembly qualified name as the identity of classes.

Types Resolution Less Restricted Mode

Sometimes, there is a need to have less restricted types resolution mode. E.g. you created database using application Prog A, but you wanted without any dependencies to it, use the database with application Prog B. You can copy classes, and having the same namespaces and class names guarantees that database will work with both applications.

In this mode, Types are identified by namespace and class name, or by namespace, class name and different assembly than you have specified in assembly qualified name of the class.

The second option is always turned on, when first one cannot resolve the type. E.g., if assembly version was changed then you cannot use the same assembly qualified name to resolve class. But the less restricted mode comes to help and you can still use all of your stored objects even after changing the version of assembly, or after moving classes between different assemblies.

4.17 Silverlight and Windows Phone limitations

Actual version of NDatabase for Silverlight and Windows Phone platform has some limitations (supported up to NDatabase 3.4).

Silverlight does not support reflection on private fields.

- That's mean, you cannot store objects of classes which are containing non public fields.

© 2014 Jacek Spolnik

- You cannot store instances of classes which are containing public properties (public properties are generating private backing fields which are responsible for storing properties data).
- You cannot use properties in Linq to NDatabase

Sample class which could be successfully stored:

```
public class Person
{
   public string Name;
   public int Age;
}
```

Limited Internal storage quota

- As in samples, you need to increase IsolatedStorage calling a method from UI.

Restricted types resolution

- You could work only in normal mode regarding types resolution (see page).

5 Examples

Here you can find several samples, which are showing NDatabase in action:
- Basic sample - creating simple structure of objects, storage, updating and deleting of objects.
- Use NDatabase with LinqPad - Northwind sample.

5.1 Basic Sample

- The purpose of this sample is to show how to work with NDatabase as the storage solution in the simple case.
- After that you could successfully store your objects into NDatabase and then work with them whenever you want.

Preparing domain objects for our sample

We need some object structure for presenting work on NDatabase. Our data model will define warriors which could hold two items. Items could be field or sword.

Items

In our data model we have two kinds of items: armor and weapon. Both contain attributes: attack and defense which will modify warrior attributes.

The base interface for all items defines two attributes:

```
public interface IItem
{
    int Attack { get; }
    int Defense { get; }
}
```

Then we have two interfaces describing the type of the item:

© 2014 Jacek Spolnik

```
public interface IArmor : IItem
{
}

public interface IWeapon : IItem
{
}
```

Implementations of above interfaces:

© 2014 Jacek Spolnik

```
public sealed class Field : IArmor
{
    private readonly int _defense;

    public Field(int value)
    {
        _defense = value;
    }

    #region IArmor Members

    public int Defense
    {
        get { return _defense; }
    }

    public int Attack
    {
        get { return 0; }
    }

    #endregion

    public override string ToString()
    {
        return string.Format("Field +{0}D", Defense);
    }
}

public sealed class Sword : IWeapon
{
    private readonly int _attack;

    public Sword(int value)
    {
        _attack = value;
    }

    #region IWeapon Members

    public int Attack
    {
        get { return _attack; }
    }

    public int Defense
    {
        get { return 0; }
    }

    #endregion

    public override string ToString()
    {
        return string.Format("Sword +{0}A", Attack);
    }
}
```

The initial state of warrior doesn't contain the items. So we need to mark that there is no item there. To do that we are using class NoItem:

© 2014 Jacek Spolnik

```
public sealed class NoItem : IItem
{
    public static readonly IItem Instance = new NoItem();

    private NoItem()
    {
    }

    #region IItem Members

    public int Attack
    {
        get { return 0; }
    }

    public int Defense
    {
        get { return 0; }
    }

    #endregion

    public override string ToString()
    {
        return "NoItem";
    }
}
```

Warrior

Warrior is defined by IHero interface:

```
public interface IHero
{
    IItem RightHand { set; }
    IItem LeftHand { set; }

    int Attack { get; }
    int Defense { get; }

    int Level { get; }
    string Name { get; }
}
```

This interface contains two items. One is hold in left hand and the second in the right hand. Additionally, hero is described by his level, name and two attributes: attack and defense. Attack and defense attributes are modified by items which hero has assigned.

The implementation of IHero interface:

© 2014 Jacek Spolnik

```
public sealed class Warrior : IHero
{
    private readonly string _name;
    private int _attack;
    private int _defense;

    public Warrior(string name)
    {
        if (string.IsNullOrEmpty(name))
            throw new ArgumentNullException("name",
                                    "Name of warrior cannot be empty.");

        _name = name;
        RightHand = NoItem.Instance;
        LeftHand = NoItem.Instance;

        Attack = 3;
        Defense = 3;
    }

    #region IHero Members

    public IItem RightHand { private get; set; }
    public IItem LeftHand { private get; set; }

    public int Attack
    {
        get { return _attack + RightHand.Attack + LeftHand.Attack; }
        private set { _attack = value; }
    }

    public int Defense
    {
        get { return _defense + RightHand.Defense + LeftHand.Defense; }
        private set { _defense = value; }
    }

    public int Level
    {
        get { return 1; }
    }

    public string Name
    {
        get { return _name; }
    }

    #endregion

    public override string ToString()
    {
        return string.Format("[{0}]: RH: {1}, LH: {2}, Att: {3}, Def: {4}, Lvl: {5}",
                             Name, RightHand, LeftHand, Attack, Defense, Level);
    }
}
```

Storing objects

Firstly, we create the two warrior objects with a different names.

© 2014 Jacek Spolnik

```
var warrior1 = new Warrior("Warrior 1");
var warrior2 = new Warrior("Warrior 2");
```

To store the objects we need to open our database with parameter as the name of our db file. Our db file name is defined by constant string:

```
private const string DBName = "game.ndb";
```

Now we could open our database and store prepared two objects.

```
using (var odb2 = OdbFactory.Open(DBName))
{
    odb2.Store(warrior1);
    odb2.Store(warrior2);
}
```

Updating them

Now, is the time for preparing our equipment which could be used by our powerful warriors.

```
var sword1 = new Sword(5);
var sword2 = new Sword(3);

var field1 = new Field(3);
var field2 = new Field(5);
```

Now we could open our database and retrieve already stored objects. Then we could use prepared items to update warriors, and then store them back into database.

For the case when you are opening database with the same name as for the first time, you can use OdbFactory.OpenLast() method to do that easier.

```
using (var odb = OdbFactory.OpenLast())
{
    IList<Warrior> warriors = odb.QueryAndExecute<Warrior>().ToList();

    warriors[0].RightHand = sword1;
    warriors[0].LeftHand = field1;

    warriors[1].RightHand = sword2;
    warriors[1].LeftHand = field2;

    odb.Store(warriors[0]);
    odb.Store(warriors[1]);
}
```

Now our database contains updated warriors, and we could perform some action on them.

Querying and displaying items from database

Now we could display items which are in the database. To do that we are opening db, then we are querying for items and finally, we are displaying the retrieved items.

© 2014 Jacek Spolnik

```
using (var odb1 = OdbFactory.OpenLast())
{
    IList<IItem> items = odb1.QueryAndExecute<IItem>().ToList();

    foreach (var item in items)
        Console.WriteLine(item);
}
```

The output for that is:

```
NoItem
Sword +5A
Sword +3A
Field +3D
Field +5D
```

Finally, we are querying for warriors and displaying them in the same way as for items. After that, we will remove both objects from db.

```
using (var odb1 = OdbFactory.OpenLast())
{
    IList<Warrior> warriors = odb1.QueryAndExecute<Warrior>().ToList();

    foreach (var warrior in warriors)
        Console.WriteLine(warrior);

    Console.WriteLine("Remove warriors");

    odb1.Delete(warriors[0]);
    odb1.Delete(warriors[1]);
}
```

The output for that is:

```
[Warrior 1]: RH: Sword +5A, LH: Field +3D, Att: 8, Def: 6, Lvl: 1
[Warrior 2]: RH: Sword +3A, LH: Field +5D, Att: 6, Def: 8, Lvl: 1
```

Now are database should have no warrior objects. To check that we will query for warriors again, and we will check the count of returned items.

```
int count;
using (var odb = OdbFactory.OpenLast())
    count = odb.QueryAndExecute<Warrior>().Count;

Console.WriteLine("Warriors count: {0}", count);
```

And the final output is:

```
Warriors count: 0
```

© 2014 Jacek Spolnik

5.2 LinqPad for NDatabase - Northwind

To see more about LinqPad, check LinqPad support page.

Prerequisites

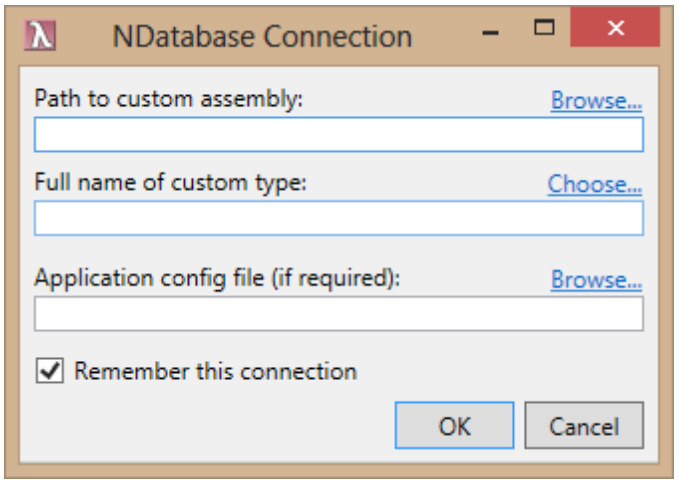
- See how to install driver: LinqPad installation page.
- Download northwind.ndb
- Download NDatabase_LinqPad_Northwind_Sample.zip

Configure sample

1. Unpack somewhere NDatabase_LinqPad_Northwind_Sample.zip.
2. NDatabase.Northwind.Domain.dll contains schema classes, which were used to store objects and additionally, it contains typed data context NDatabase.Northwind.Domain.NDatabaseNorthwindDataContext which will allow LinqPad to query NDatabase norhtwind db
3. Set in NDatabase.Northwind.Domain.dll.config path to northwind.ndb (full path is needed).

Add new connection

1. Open LinqPad, then click “Add Connection”.
2. Choose NDatabase static driver.
3. Click "Next", and use below settings:
   a. set path to NDatabase.Northwind.TypedDataContext.dll,
   b. choose NDatabase.Northwind.TypedDataContext.NDatabaseNorthwindDataContext type,
   c. set path to NDatabase.Northwind.TypedDataContext.dll.config.

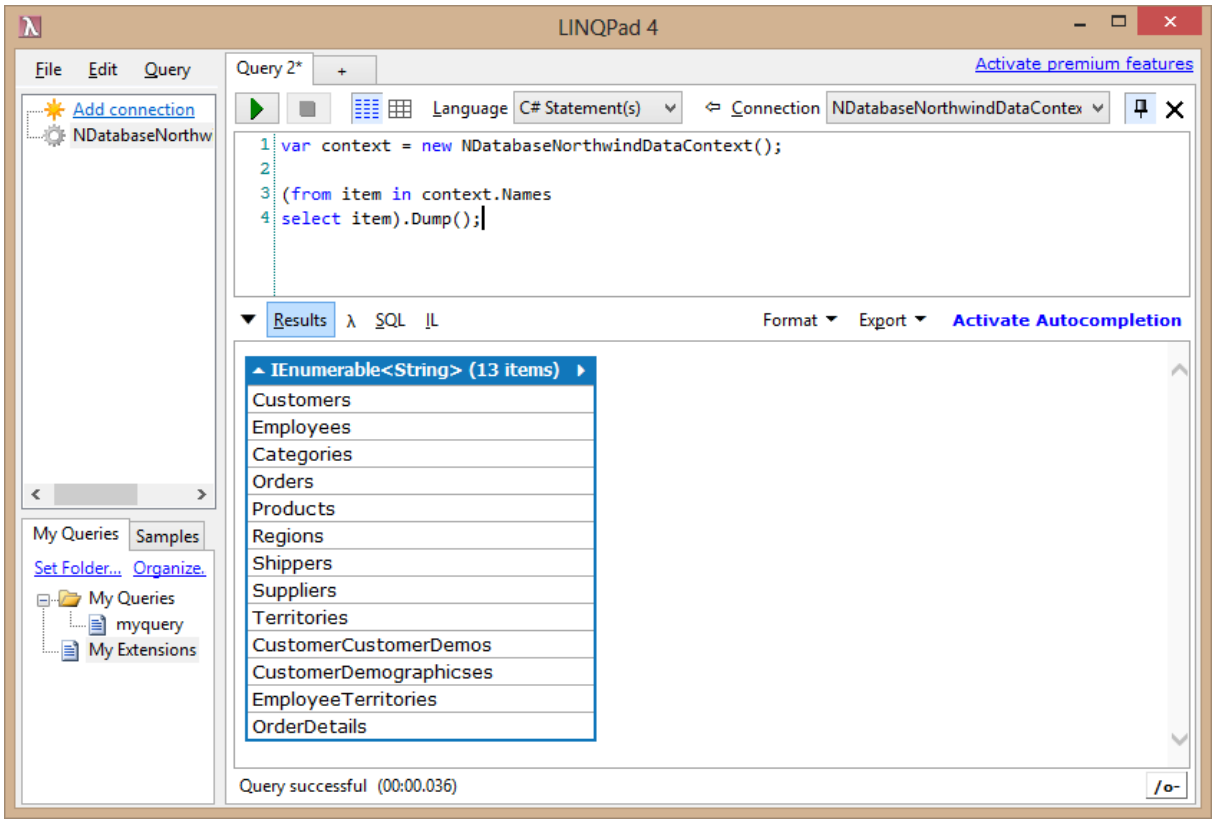


After clicking "OK", connection is ready to use.

Run queries

Run the command from screen. You can replace "Names" with any value from the below result:

© 2014 Jacek Spolnik



Typed data context prepared for Northwind NDatabase

© 2014 Jacek Spolnik

```
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using NDatabase.Northwind.Domain;
using NDatabase;

namespace NDatabase.Northwind.TypedDataContext
{
    public class NDatabaseNorthwindDataContext
    {
        static NDatabaseNorthwindDataContext()
        {
            if (ConfigurationManager.AppSettings.AllKeys.Contains("DbFilePath"))
            {
                DbName = ConfigurationManager.AppSettings["DbFilePath"];
            }
            else
            {
                throw new ConfigurationErrorsException(
                    "You are missing key 'DbFilePath' in your <appSettings>. Key has to contain va
            }
        }

        private static readonly string DbName;

        private static readonly IList<string> PropertyNames = new List<string>
                                                              {
                                                                  "Customers",
                                                                  "Employees",
                                                                  "Categories",
                                                                  "Orders",
                                                                  "Products",
                                                                  "Regions",
                                                                  "Shippers",
                                                                  "Suppliers",
                                                                  "Territories",
                                                                  "CustomerCustomerDemos",
                                                                  "CustomerDemographicses",
                                                                  "EmployeeTerritories",
                                                                  "OrderDetails"
                                                              };

        public IEnumerable<string> Names
        {
            get { return PropertyNames; }
        }

        public IEnumerable<Customer> Customers
        {
            get
            {
                IList<Customer> result;
                using (var odb = OdbFactory.Open(DbName))
                {
                    result = odb.Query<Customer>().Execute<Customer>().ToList();
                }
                return result;
            }
        }

        public IEnumerable<Employee> Employees
```

```
        {
            get
            {
                IList<Employee> result;
                using (var odb = OdbFactory.Open(DbName))
                {
                    result = odb.Query<Employee>().Execute<Employee>().ToList();
                }
                return result;
            }
        }

        public IEnumerable<Category> Categories
        {
            get
            {
                IList<Category> result;
                using (var odb = OdbFactory.Open(DbName))
                {
                    result = odb.Query<Category>().Execute<Category>().ToList();
                }
                return result;
            }
        }

        public IEnumerable<Order> Orders
        {
            get
            {
                IList<Order> result;
                using (var odb = OdbFactory.Open(DbName))
                {
                    result = odb.Query<Order>().Execute<Order>().ToList();
                }
                return result;
            }
        }

        public IEnumerable<Product> Products
        {
            get
            {
                IList<Product> result;
                using (var odb = OdbFactory.Open(DbName))
                {
                    result = odb.Query<Product>().Execute<Product>().ToList();
                }
                return result;
            }
        }

        public IEnumerable<Region> Regions
        {
            get
            {
                IList<Region> result;
                using (var odb = OdbFactory.Open(DbName))
                {
                    result = odb.Query<Region>().Execute<Region>().ToList();
                }
                return result;
```

© 2014 Jacek Spolnik

```
        }
    }

    public IEnumerable<Shipper> Shippers
    {
        get
        {
            IList<Shipper> result;
            using (var odb = OdbFactory.Open(DbName))
            {
                result = odb.Query<Shipper>().Execute<Shipper>().ToList();
            }
            return result;
        }
    }

    public IEnumerable<Supplier> Suppliers
    {
        get
        {
            IList<Supplier> result;
            using (var odb = OdbFactory.Open(DbName))
            {
                result = odb.Query<Supplier>().Execute<Supplier>().ToList();
            }
            return result;
        }
    }

    public IEnumerable<Territory> Territories
    {
        get
        {
            IList<Territory> result;
            using (var odb = OdbFactory.Open(DbName))
            {
                result = odb.Query<Territory>().Execute<Territory>().ToList();
            }
            return result;
        }
    }

    public IEnumerable<CustomerCustomerDemo> CustomerCustomerDemos
    {
        get
        {
            IList<CustomerCustomerDemo> result;
            using (var odb = OdbFactory.Open(DbName))
            {
                result = odb.Query<CustomerCustomerDemo>().Execute<CustomerCustomerDemo>().ToI
            }
            return result;
        }
    }

    public IEnumerable<CustomerDemographics> CustomerDemographicses
    {
        get
        {
            IList<CustomerDemographics> result;
            using (var odb = OdbFactory.Open(DbName))
```

```
                {
                    result = odb.Query<CustomerDemographics>().Execute<CustomerDemographics>().ToI
                }
                return result;
            }
        }

        public IEnumerable<EmployeeTerritory> EmployeeTerritories
        {
            get
            {
                IList<EmployeeTerritory> result;
                using (var odb = OdbFactory.Open(DbName))
                {
                    result = odb.Query<EmployeeTerritory>().Execute<EmployeeTerritory>().ToList();
                }
                return result;
            }
        }

        public IEnumerable<OrderDetail> OrderDetails
        {
            get
            {
                IList<OrderDetail> result;
                using (var odb = OdbFactory.Open(DbName))
                {
                    result = odb.Query<OrderDetail>().Execute<OrderDetail>().ToList();
                }
                return result;
            }
        }
    }
}
```

6 Internal Architecture

Here you can find pages describing, how internally NDatabase is working, which assumptions it is following, etc:

- OIDs
- Layers
- File Format
- Execution Mode

6.1 OIDs

- OID stands for Object ID.
- In NDatabase, every entity (class or object) has an OID.
- An OID uniquely identifies an entity in the NDatabase.
- NDatabase has special blocks to keep associations between the OID and the physical position of the entity in the database file. The OID allows a direct access to the object with a reduced number of IO operations.
- All pointers in the database use OIDs. When an object has a relation with another, the relation is stored using the OID object, for example:
  - If a user has an attribute profile (with OID 15), the user object will be stored as having a relation with the OID 15.

© 2014 Jacek Spolnik

Implementation

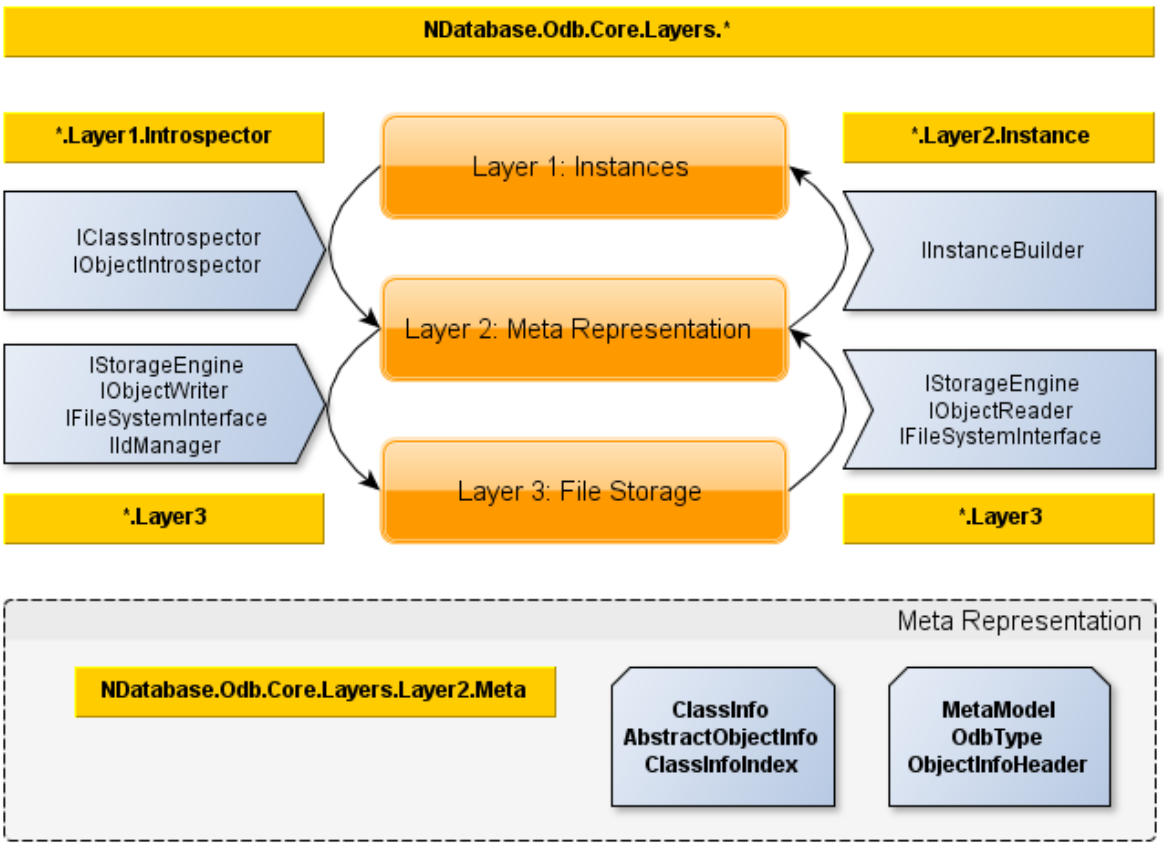
- The OID is represented by OID interface. Default implementations are internal classes ClassOID and ObjectOID.
- OIDs are managed by the IdManager internal class, which implements internal IIdManager interface.

```
public interface OID : IComparable
{
    long ObjectId { get; }
}
```

6.2 Layers

NDatabase engine contains 3 layers of abstraction to manage the data:
- Layer 1: Instance Layer
- Layer 2: Meta representation of objects and classes
- Layer 3: The physical storage

Layers Big Picture



Layer 1: The instance layer

- This layer works with Instances.

© 2014 Jacek Spolnik

- This is the entry point of NDatabase.
- It is the user interface to NDatabase.

Implementation (Layer 1 -> Layer 2):
- ObjectIntrospector - uses reflection to transform an object into a NonNativeObjectInfo meta representation.
- ClassIntrospector - uses reflection to transform a class into a ClassInfo meta representation.

Implementation (Layer 2 -> Layer 1):
- InstanceBuilder - transforms a NonNativeObjectInfo into an object. It is used when retrieving objects.

Layer 2: The meta representation

- This layer is a Meta Representation layer.
- It holds all the data and definition in a language independent way.
- Main classes are ClassInfo (Meta representation of a class), NonNativeObjectInfo (Meta representation of an instance), NativeObjectInfo (Meta representation of a native value, like int, long, String) and other minor classes.

Layer 3: The file storage

NDatabase stores all data in a unique binary file. This file will contain:
- The database meta-model : All the classes already stored in the base.
- The instances.
- The indexes.
- And other things, for the details please refer to documentation page about NDatabase file format.

Implementation (Layer 2 to Layer 3):
- ObjectWriter - writes a NonNativeObjectInfo to the database file.

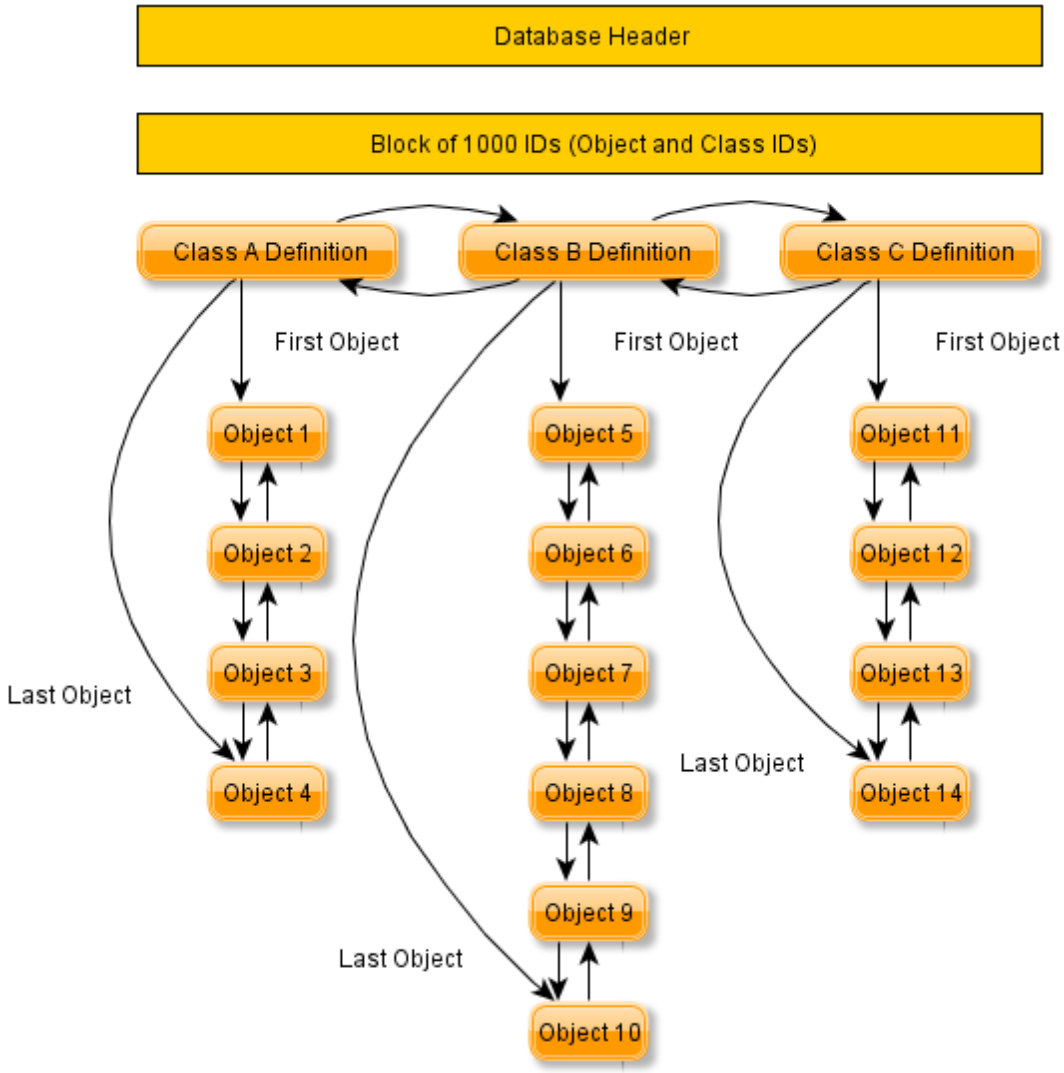
Implementation (Layer 3 to Layer 2):
- ObjectReader - reads the database file and creates a NonNativeObjectInfo.

6.3 File Format

NDatabase stores all its data in a single file:
- Meta-model : classes that are stored in the object base,
- Objects thats are stored,
- Indexes.

File format big picture

© 2014 Jacek Spolnik



NDatabase file format has the following block types:

- Database Header Block
- ID Block
- Class Info Block
- Object Block
- String Block

Indexes are stored in the database as normal objects using the ClassInfoIndex class. This class is marked as a system class.

Internal storage

In the current version of NDatabase, entities are stored in the database file as two ways linked list.

Let's take an example to understand that:

An object which is managed by a NonNativeObjectInfo (Layer 2) object contains the data of the objects,

© 2014 Jacek Spolnik

the OID of the its class (ClassInfo) and some pointers:
- pointer to the next object of the same type,
- pointer to the previous object of the same type,

This pointers allow the navigation from an object to other. Like a two ways linked list. This is simple way to handle object navigation but have some drawbacks for the Isolation property of ACID: this issue will be fixed in some future version.

6.3.1 Database Header Block

No	Field	Description	Position	Size
1	Version	The NDatabase File format version	0	1 int
2	Database ID	A unique identifier of the NDatabase database file	4	4 long
3	Last Transaction sequential ID	The last Transaction Sequential ID	36	2 long
4	Number of classes	The number of classes in meta model	36+2*8=52	1 long
5	First ClassInfo OID	The OID of the first ClassInfo of the meta model	60	1 long
6	Last ODB Close status	To check last close status : 1 is ok, 0 is not	68	1 boolean
7	Empty Space	Space which could be used in the future version of NDatabase	69	120 bytes
8	Database String Encoding	The Database String Encoding, default is UTF-8	189	50 bytes + 8 bytes for string size = 58 bytes
9	Current block id position	The current id block to be used	247	1 long
10	First ID Block	The first block of ids	255	1000 bytes

Header size = 255
Total header size, including first id block of 1000 entries = 255 + 18034 = 18289

6.3.2 ID Block

The ID Block is used to store all ids of objects and classes. In the ID Block, the following data are stored:
- The Object/class ID
- The OID type (Object or class)
- The physical position of the object with the OID
- An indicator to say if OID is still active. For example deleted objects are marks as deleted

The NDatabase header holds the position of the first ID block (field 10).

© 2014 Jacek Spolnik

The ID Block can contains up to 1000 OIDs definitions. It has a fixed size :
- ID Block size = size of header + 1000 * size of a repetition = 34 + 1000 * 18 = 18034 bytes

The ID Blocks are managed by the IdManager internal class. The actual writing into database file is done by the another internal class, ObjectWriter.

ID Block Header

No	Field	Description	Position	Size
1	Block size	The block size	0	1 int
2	Block type	20 (BlockTypeIds)	4	1 byte
3	Block status	To indicate if block is full, 1 = no, 2 = full	5	1 byte
4	Prev block position	The previous ID block position	6	1 long
5	Next block position	The next ID block position	14	1 long
6	Block number	The block number	22	1 int
7	Max id	Max id already allocated in this block	26	1 long

ID Block Repetition

No	Field	Description	Size
1	ID type	To indicate if it is an ID of object or class	1 byte
2	OID	The Object ID	1 long
3	Status	The status of the OID, 1 = Active, 2 = Deleted (ID Status)	1 byte
4	Position	The physical position of the object with this OID	1 long

6.3.3 Class Info Block

The meta model of the database is stored using ClassInfo which is a meta-representation of a native class.

Class Info Header

No	Field	Description	Position	Size
1	Block size	The block size	0	1 int
2	Block type	1 (BlockTypeClassHeader)	4	1 byte
3	Class type	System class = 1, User class = 2	5	1 byte
4	Class id	The id of the class	6	1 long
5	Prev class info OID	The OID of the previous Class Info	14	1 long
6	Next class info OID	The OID of the next Class Info	22	1 long
7	Number of objects	The number of objects of this class	30	1 long
8	First object OID	The OID of the first object of this class	38	1 long
9	Last Object OID	The OID of the last object of this class	46	1 long
10	Full class name (with namespace)	The name of the class	54	variable
11	Class attribute definition position	The position where to find the class attributes definition	-	1 long

© 2014 Jacek Spolnik

Class Info Attributes

When the attribute is a Native Object: non-array

No	Field	Description	Position	Size
1	Is Native	1 if native	0	1 byte
2	Odb Type id	OdbType	1	1 int
3	Name	Attribute name	5	variable
4	Is used in index	If attribute is used in an index	-	1 byte

When the attribute is a Native Object: array

No	Field	Description	Position	Size
1	Is Native	1 if native	0	1 byte
2	Odb Type id	OdbType	1	1 int
3	Sub Type id	The Odb Type of the array elements	5	1 int
4	Class Info OID	The class info OID if sub type is a NonNative Object. If not, this field does not exist	9	1 int
5	Name	Attribute name	9 or 13	variable
6	Is used in index	If attribute is used in an index	-	1 byte

When the attribute is a Non Native Object

No	Field	Description	Position	Size
1	Is Native	It is not native: 0	0	1 byte
2	Class Info OID	The class info OID of the attribute	1	1 long
3	Name	Attribute name	9	variable
4	Is used in index	If attribute is used in an index	-	1 byte

6.3.4 Object Block

No	Field	Description	Position	Size
1	Block size	The size of the block	0	1 int
2	Block type	4 (BlockTypeNonNativeObject)	1	1 byte
3	OID	The Object ID	5	1 long
4	Class Info OID	The OID of the class of the object	13	1 long
5	Previous Object OID	OID of the previous object	21	1 long
6	Next Object OID	OID if the next Object	29	1 long
7	Creation Date	The Object creation date	37	1 long
8	Update Date	The Object update date	45	1 long
9	Version Number	Object version number	53	1 int
10	Object Reference	Not used yet. If this objects Is being referenced by others,then this field will be greater > 0.It will point to an internal object of type ObjectReference that will have details on the references. This is to enable object integrity.	57	1 long
11	External synchronization indicator	Not used yet. True if this object have been synchronized with main database, else false	65	1 bool

© 2014 Jacek Spolnik

12	Number of attributes	Number of attributes of the object	66	1 int

Then write the attribute data.

First write a block of NbAttributes elements. Each element contains the id of the attribute (int) and the OID or position of the value of the attribute.

No	Field	Description	Position	Size
1	Attribute ID	This id is the ID of the attribute that was defined in the ClassInfo	0	1 int
2	Attribute value OID	How/where to find the value of the attribute. For native objects it is the physical position, for non native objects it is the OID (In this case, the OID, long value, is stored as a negative number to differ from position)	4	1 long

Then writes the values of the attributes. Values of Native Objects are written in place. Non Native Objects are written at the end of the file.

Arrays are written in the object block as native objects :

Arrays

No	Field	Description	Position	Size
1	Block type	The block type	-	1 int
2	Block size	The block size	-	1 byte
3	Odb Type ID	OdbType	-	1 int
4	Is Null?	To indicate if object is null	-	1 bool
5	Class Name	The real Collection Class Name	-	variable
6	Size	Size of the list	-	1 long
7	Block of element identification	A block of OID of the elements of the list. If an element is native (ex: a string), the its position is stored instead of OID	-	block of long

Native Attributes

No	Field	Description	Position	Size
1	Block size	The block size, fix = 9	-	1 int
2	Block type	The block type	-	1 byte
3	Attribute type	The odb type id	-	1 int
4	Is null?	To indicate if object is null	-	1 bool
5	The attribute value	The bytes of the attribute value	-	variable

6.3.5 String Block

Strings have a specific way to be written to file. A String in NDatabase have 2 size attributes:
- the max size that have been reserved for the string (to enable storing a bigger string in the same place)
- the real string size

The bytes that represent the string are encoded using UTF-8.

No	Field	Description	Position	Size

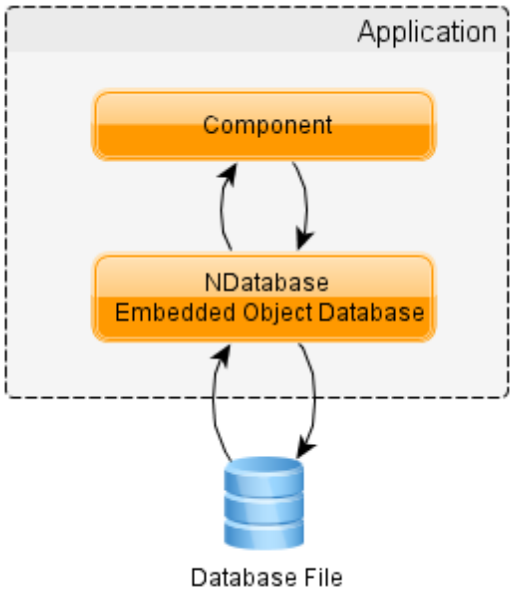
© 2014 Jacek Spolnik

1	Total size	Total String size	-	1 int
2	Real size	Real String size	-	1 int
3	The string	The bytes of the string	-	variable

The encoding and decoding are implemented in ByteArrayConverter internal class.

6.4 Execution Mode

NDatabase can be used only in local mode in actual version.



- Local mode is used when one want to run NDatabase as an embedded database.
- NDatabase runs within the application as the part of it.
- This is the easiest and fastest way to execute Object Database.
- Database could be used by many threads, or by many processes with many threads.

7 Project Origin

Firstly, I want to thanks Olivier Smadja for his work on NeoDatis java version. He did a great job and based on that I could create a NDatabase project.

Project derives from NeoDatis ODB project. The .net version of this project was never released (as the stable version). The work is based on NeoDatis.Net 1.9-beta6 code, which you can find there. Documentation about NeoDatis could be found here.

NDatabase was radically changed in comparison to NeoDatis.Net code base, but still the core architecture is living there. You can find more documentation about NeoDatis here.

Changes in comparison to NeoDatis (NDatabase 1.0.4):
- lot of bug fixes
- fixing & updating unit tests
- using more & more power of C# instead of using generic solutions between Java & C#

© 2014 Jacek Spolnik

- remove unwanted things like Client/Server functionality (NDatabase is focused on embedded solution).
- updated documentation which fits to the C# world
- all queries are polymorphic
- API improvements
- performance improvements
- and more and more, with the time the amount of changes will increase

You need to remember, that this both solutions are not compatible and the File Format could be different for both.

8 FAQ

Do my objects need to implement a specific NDatabase interface to persist them ?
- No, there is no restriction on objects to be stored.

Do my class must to have Serializable attribute ?
- No, there is no such a requirement.

Do NDatabase support circular references ?
- Yes, circular references are supported.

Can I only store public fields ?
- No, you can store all fields

Can I store readonly private fields ?
- Yes, you can.

Can I query for private or internal fields ?
- Yes, you can, but you need to use SODA queries. LINQ to NDatabase, because of its nature is not supporting that.

Is it possible to ignore some fields in a class, so that they don't get stored ?
- Yes, you need to use [NonPersistent] attribute. Here you can find information about this attribute.

Why my class is not recognized when I moved it to the another assembly ?
- That's happen because every type is identified by <namespace>.<class_name>, <assembly_name> in normal mode. If you changed the assembly, that's mean you have a new type. To resolve that, you need to use Types Resolution less restricted mode. Check portability page.

9 Reference

The Public API is prepared based on xml documentation of NDatabase project (prepared by Sandcastle).

9.1 NDatabase Namespace

Classes

	Class	Description
	OdbFactory	The NDatabase Factory to open new instance of local odb.

© 2014 Jacek Spolnik

	OIDFactory	Factory class to create OIDs

Interfaces

	Interface	Description
	ILogger	Base interface for creacting custom logger

9.1.1 ILogger Interface

Base interface for creacting custom logger

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface ILogger
```

C#

```
public interface ILogger
```

Visual C++

```
public interface class ILogger
```

JavaScript

```
NDatabase.ILogger = function();
NDatabase.ILogger.createInterface('NDatabase.ILogger');
```

See Also
ILogger Members
NDatabase Namespace

9.1.1.1 ILogger Members

The ILogger type exposes the following members.

Methods

	Name	Description
	Debug	Log message with debug level
	Error	Log message with error level
	Info	Log message with info level
	Warning	Log message with warn level

See Also
ILogger Interface
NDatabase Namespace

9.1.1.2 ILogger Methods

The ILogger type exposes the following members.

Methods

© 2014 Jacek Spolnik

	Name	Description
	Debug	Log message with debug level
	Error	Log message with error level
	Info	Log message with info level
	Warning	Log message with warn level

See Also
ILogger Interface
NDatabase Namespace

9.1.1.2.1 Debug Method

Log message with debug level

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub Debug ( _
 message As String _
)
```

C#

```
void Debug(
 string message
)
```

Visual C++

```
void Debug(
 String^ message
)
```

JavaScript

```
function debug(message);
```

Parameters

message
Type: System.
Mssage to log

See Also
ILogger Interface
NDatabase Namespace

9.1.1.2.2 Error Method

Log message with error level

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub Error ( _
```

© 2014 Jacek Spolnik

```
 message As String _
)
```

C#

```
void Error(
 string message
)
```

Visual C++

```
void Error(
 String^ message
)
```

JavaScript

```
function error(message);
```

Parameters

message
Type: System.
Mssage to log

See Also

ILogger Interface
NDatabase Namespace

9.1.1.2.3 Info Method

Log message with info level

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub Info ( _
 message As String _
)
```

C#

```
void Info(
 string message
)
```

Visual C++

```
void Info(
 String^ message
)
```

JavaScript

```
function info(message);
```

Parameters

message
Type: System.
Mssage to log

See Also

ILogger Interface
NDatabase Namespace

9.1.1.2.4 Warning Method

Log message with warn level

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub Warning ( _
 message As String _
)
```

C#

```
void Warning(
 string message
)
```

Visual C++

```
void Warning(
 String^ message
)
```

JavaScript

```
function warning(message);
```

Parameters

message
Type: System.
Mssage to log

See Also
ILogger Interface
NDatabase Namespace

9.1.2 OdbFactory Class

The NDatabase Factory to open new instance of local odb.

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public NotInheritable Class OdbFactory
```

C#

```
public static class OdbFactory
```

Visual C++

```
public ref class OdbFactory abstract sealed
```

JavaScript

```
NDatabase.OdbFactory = function();
```

```
Type.createClass(
 'NDatabase.OdbFactory');
```

Inheritance Hierarchy

System.
NDatabase.

See Also
OdbFactory Members
NDatabase Namespace

9.1.2.1 OdbFactory Members

The OdbFactory type exposes the following members.

Methods

	Name	Description
	Delete	Deletes the specified file name.
	Open	Opens the database instance with the specified file name.
	OpenInMemory	Opens a database in the In-Memory mode.
	OpenLast	Opens the database instance with the last given name.

See Also
OdbFactory Class
NDatabase Namespace

9.1.2.2 OdbFactory Methods

The OdbFactory type exposes the following members.

Methods

	Name	Description
	Delete	Deletes the specified file name.
	Open	Opens the database instance with the specified file name.
	OpenInMemory	Opens a database in the In-Memory mode.
	OpenLast	Opens the database instance with the last given name.

See Also
OdbFactory Class
NDatabase Namespace

9.1.2.2.1 Delete Method

Deletes the specified file name.

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

© 2014 Jacek Spolnik

Syntax

Visual Basic

```
Public Shared Sub Delete ( _
 fileName As String _
)
```

C#

```
public static void Delete(
 string fileName
)
```

Visual C++

```
public:
static void Delete(
 String^ fileName
)
```

JavaScript

```
NDatabase.OdbFactory.delete = function(fileName);
```

Parameters

fileName

Type: System.
Name of the file.

See Also

OdbFactory Class
NDatabase Namespace

9.1.2.2.2 Open Method

Opens the database instance with the specified file name.

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Function Open ( _
 fileName As String _
) As IOdb
```

C#

```
public static IOdb Open(
 string fileName
)
```

Visual C++

```
public:
static IOdb^ Open(
 String^ fileName
)
```

JavaScript

```
NDatabase.OdbFactory.open = function(fileName);
```

© 2014 Jacek Spolnik

Parameters

fileName

Type: System.
Name of the file.

Return Value

IOdb.

See Also

OdbFactory Class
NDatabase Namespace

9.1.2.2.3 OpenInMemory Method

Opens a database in the In-Memory mode.

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Function OpenInMemory As IOdb
```

C#

```
public static IOdb OpenInMemory()
```

Visual C++

```
public:
static IOdb^ OpenInMemory()
```

JavaScript

```
NDatabase.OdbFactory.openInMemory = function();
```

Return Value

IOdb implementation.

See Also

OdbFactory Class
NDatabase Namespace

9.1.2.2.4 OpenLast Method

Opens the database instance with the last given name.

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Function OpenLast As IOdb
```

C#

```
public static IOdb OpenLast()
```

Visual C++

```
public:
```

© 2014 Jacek Spolnik

```
static IOdb^ OpenLast()
```

JavaScript

```
NDatabase.OdbFactory.openLast = function();
```

Return Value
IOdb.

See Also
OdbFactory Class
NDatabase Namespace

9.1.3 OIDFactory Class

Factory class to create OIDs

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public NotInheritable Class OIDFactory
```

C#

```
public static class OIDFactory
```

Visual C++

```
public ref class OIDFactory abstract sealed
```

JavaScript

```
NDatabase.OIDFactory = function();
Type.createClass(
 'NDatabase.OIDFactory');
```

Inheritance Hierarchy

System.
NDatabase.

See Also
OIDFactory Members
NDatabase Namespace

9.1.3.1 OIDFactory Members

The OIDFactory type exposes the following members.

Methods

	Name	Description
	BuildClassOID	Build class oid based on long number
	BuildObjectOID	Build object oid based on long number

See Also
OIDFactory Class

© 2014 Jacek Spolnik

NDatabase Namespace

9.1.3.2 OIDFactory Methods

The OIDFactory type exposes the following members.

Methods

	Name	Description
	BuildClassOID	Build class oid based on long number
	BuildObjectOID	Build object oid based on long number

See Also
OIDFactory Class
NDatabase Namespace

9.1.3.2.1 BuildClassOID Method

Build class oid based on long number

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Function BuildClassOID ( _
 oid As Long _
) As OID
```

C#

```
public static OID BuildClassOID(
 long oid
)
```

Visual C++

```
public:
static OID^ BuildClassOID(
 long long oid
)
```

JavaScript

```
NDatabase.OIDFactory.buildClassOID = function(oid);
```

Parameters

oid
Type: System.
long number as the base for OID

Return Value
Newly created OID

See Also
OIDFactory Class
NDatabase Namespace

© 2014 Jacek Spolnik

9.1.3.2.2 BuildObjectOID Method

Build object oid based on long number

Namespace: NDatabase
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Function BuildObjectOID ( _
 oid As Long _
) As OID
```

C#

```
public static OID BuildObjectOID(
 long oid
)
```

Visual C++

```
public:
static OID^ BuildObjectOID(
 long long oid
)
```

JavaScript

```
NDatabase.OIDFactory.buildObjectOID = function(oid);
```

Parameters

oid

Type: System.
long number as the base for OID

Return Value

Newly created OID

See Also

OIDFactory Class
NDatabase Namespace

9.2 NDatabase.Api Namespace

Classes

	Class	Description
	CascadeDeleteAttribute	Use when you want to do a cascade delete on field object.
	NonPersistentAttribute	Use when you don't want to serialize the field.
	OdbConfiguration	The main NDatabase ODB Configuration class.
	OIDAttribute	Use when you want to enrich your class with OID. You can apply it on fields of type: long or OID.
	OrderByConstants	Constants used for ordering queries and creating ordered collection iterators

Interfaces

	Interface	Description

© 2014 Jacek Spolnik

	IDatabaseId	Database identification
	IExternalOID	External OID, which contains database id
	IIndexManager	Index Manager - allows to give access to class level configuration like adding an index, checking if index exists, rebuilding an index,...
	IObjectRepresentation	used to give the user an instance of an object representation, level2.
	IObjectSet()	query resultset. The Copy Code `ObjectSet` interface serves as a cursor to iterate through a set of objects retrieved by a query.
	IObjectValues	Interface that will be implemented to hold a row of a result of an Object Values Query
	IOdb	Database engine interface.
	IOdbComparable	NDatabase wrapper to the native Comparable interface
	IOdbExt	An interface to provider extended access to ODB.
	IOdbForTrigger	Database engine interface (simplified for triggers purpose).
	IRefactorManager	An interface for refactoring
	IValues	The main interface of all Object Values query results of NDatabase ODB
	OID	Object ID interface

9.2.1 CascadeDeleteAttribute Class

Use when you want to do a cascade delete on field object.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
<AttributeUsageAttribute(AttributeTargets.Field)> _
Public NotInheritable Class CascadeDeleteAttribute _
 Inherits Attribute
```

C#

```
[AttributeUsageAttribute(AttributeTargets.Field)]
public sealed class CascadeDeleteAttribute : Attribute
```

Visual C++

```
[AttributeUsageAttribute(AttributeTargets::Field)]
public ref class CascadeDeleteAttribute sealed : public Attribute
```

JavaScript

```
NDatabase.Api.CascadeDeleteAttribute = function();
Type.createClass(
 'NDatabase.Api.CascadeDeleteAttribute',
 Attribute);
```

Remarks

In such case, mark the attribute with

```
[CascadeDelete]
```

Inheritance Hierarchy

System.
System.
NDatabase.Api.

See Also
CascadeDeleteAttribute Members
NDatabase.Api Namespace

9.2.1.1 CascadeDeleteAttribute Members

The CascadeDeleteAttribute type exposes the following members.

Constructors

	Name	Description
	CascadeDeleteAttribute	

Methods

	Name	Description
	Equals	(Inherited from Attribute.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Attribute.)
	GetType	(Inherited from Object.)
	IsDefaultAttribute	(Inherited from Attribute.)
	Match	(Inherited from Attribute.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

Properties

	Name	Description
	TypeId	(Inherited from Attribute.)

Explicit Interface Implementations

	Name	Description
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)

© 2014 Jacek Spolnik

See Also
CascadeDeleteAttribute Class
NDatabase.Api Namespace

9.2.1.2 CascadeDeleteAttribute Constructor

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Sub New
```

C#

```
public CascadeDeleteAttribute()
```

Visual C++

```
public:
CascadeDeleteAttribute()
```

JavaScript

```
NDatabase.Api.CascadeDeleteAttribute = function();
```

See Also
CascadeDeleteAttribute Class
NDatabase.Api Namespace

9.2.1.3 CascadeDeleteAttribute Methods

The CascadeDeleteAttribute type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Attribute.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Attribute.)
	GetType	(Inherited from Object.)
	IsDefaultAttribute	(Inherited from Attribute.)
	Match	(Inherited from Attribute.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

Explicit Interface Implementations

	Name	Description
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)

© 2014 Jacek Spolnik

	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)

See Also
CascadeDeleteAttribute Class
NDatabase.Api Namespace

9.2.1.4 CascadeDeleteAttribute Properties

The CascadeDeleteAttribute type exposes the following members.

Properties

	Name	Description
	TypeId	(Inherited from Attribute.)

See Also
CascadeDeleteAttribute Class
NDatabase.Api Namespace

9.2.2 IDatabaseId Interface

Database identification

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface IDatabaseId
```

C#

```
public interface IDatabaseId
```

Visual C++

```
public interface class IDatabaseId
```

JavaScript

```
NDatabase.Api.IDatabaseId = function();
NDatabase.Api.IDatabaseId.createInterface('NDatabase.Api.IDatabaseId');
```

See Also
IDatabaseId Members
NDatabase.Api Namespace

9.2.2.1 IDatabaseId Members

The IDatabaseId type exposes the following members.

Methods

	Name	Description

© 2014 Jacek Spolnik

	GetIds	Long numbers identifing database

See Also
IDatabaseId Interface
NDatabase.Api Namespace

9.2.2.2 IDatabaseId Methods

The IDatabaseId type exposes the following members.

Methods

	Name	Description
	GetIds	Long numbers identifing database

See Also
IDatabaseId Interface
NDatabase.Api Namespace

9.2.2.2.1 GetIds Method

Long numbers identifing database

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function GetIds As Long()
```

C#
```
long[] GetIds()
```

Visual C++
```
array<long long>^ GetIds()
```

JavaScript
```
function getIds();
```

Return Value
Array of long numbers which identifies the database

See Also
IDatabaseId Interface
NDatabase.Api Namespace

9.2.3 IExternalOID Interface

External OID, which contains database id

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

© 2014 Jacek Spolnik

```
Public Interface IExternalOID _
 Inherits OID, IComparable(Of OID), IComparable
```

C#

```
public interface IExternalOID : OID,
 IComparable<OID>, IComparable
```

Visual C++

```
public interface class IExternalOID : OID,
 IComparable<OID^>, IComparable
```

JavaScript

```
NDatabase.Api.IExternalOID = function();
NDatabase.Api.IExternalOID.createInterface('NDatabase.Api.IExternalOID');
```

See Also
IExternalOID Members
NDatabase.Api Namespace

9.2.3.1 IExternalOID Members

The IExternalOID type exposes the following members.

Methods

	Name	Description
	CompareTo(Object)	(Inherited from IComparable.)
	CompareTo(T)	(Inherited from IComparable()
	GetDatabaseId	Get database id

Properties

	Name	Description
	ObjectId	Underlying long number - oid (Inherited from OID.)

See Also
IExternalOID Interface
NDatabase.Api Namespace

9.2.3.2 IExternalOID Methods

The IExternalOID type exposes the following members.

Methods

	Name	Description
	CompareTo(Object)	(Inherited from IComparable.)
	CompareTo(T)	(Inherited from IComparable()
	GetDatabaseId	Get database id

See Also
IExternalOID Interface
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.3.2.1 CompareTo Method

Overload List

	Name	Description
	CompareTo(Object)	(Inherited from IComparable.)
	CompareTo(T)	(Inherited from IComparable()

See Also
IExternalOID Interface
IExternalOID Members
NDatabase.Api Namespace

9.2.3.2.2 GetDatabaseId Method

Get database id

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetDatabaseId As IDatabaseId
```

C#

```
IDatabaseId GetDatabaseId()
```

Visual C++

```
IDatabaseId^ GetDatabaseId()
```

JavaScript

```
function getDatabaseId();
```

Return Value
Database Id

See Also
IExternalOID Interface
NDatabase.Api Namespace

9.2.3.3 IExternalOID Properties

The IExternalOID type exposes the following members.

Properties

	Name	Description
	ObjectId	Underlying long number - oid (Inherited from OID.)

See Also
IExternalOID Interface
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.4 IIndexManager Interface

Index Manager - allows to give access to class level configuration like adding an index, checking if index exists, rebuilding an index,...

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface IIndexManager
```

C#

```
public interface IIndexManager
```

Visual C++

```
public interface class IIndexManager
```

JavaScript

```
NDatabase.Api.IIndexManager = function();
NDatabase.Api.IIndexManager.createInterface('NDatabase.Api.IIndexManager');
```

See Also
IIndexManager Members
NDatabase.Api Namespace

9.2.4.1 IIndexManager Members

The IIndexManager type exposes the following members.

Methods

	Name	Description
	AddIndexOn	
	AddUniqueIndexOn	
	DeleteIndex	Delete existing index
	ExistIndex	To check if an index exist
	RebuildIndex	Rebuild existing index

See Also
IIndexManager Interface
NDatabase.Api Namespace

9.2.4.2 IIndexManager Methods

The IIndexManager type exposes the following members.

Methods

	Name	Description
	AddIndexOn	
	AddUniqueIndexOn	
	DeleteIndex	Delete existing index

© 2014 Jacek Spolnik

	ExistIndex	To check if an index exist
	RebuildIndex	Rebuild existing index

See Also
IIndexManager Interface
NDatabase.Api Namespace

9.2.4.2.1 AddIndexOn Method

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub AddIndexOn ( _
 indexName As String, _
 ParamArray indexFields As String() _
)
```

C#

```
void AddIndexOn(
 string indexName,
 params string[] indexFields
)
```

Visual C++

```
void AddIndexOn(
 String^ indexName,
 ... array<String^>^ indexFields
)
```

JavaScript

```
function addIndexOn(indexName, ... indexFields);
```

Parameters

indexName
Type: System.
The name of the index

indexFields
Type: .[]
The list of fields of the index. Every field needs to implement IComparable.

See Also
IIndexManager Interface
NDatabase.Api Namespace

9.2.4.2.2 AddUniqueIndexOn Method

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub AddUniqueIndexOn ( _
 indexName As String, _
```

© 2014 Jacek Spolnik

```
 ParamArray indexFields As String() _
)
```

C#

```
void AddUniqueIndexOn(
 string indexName,
 params string[] indexFields
)
```

Visual C++

```
void AddUniqueIndexOn(
 String^ indexName,
 ... array<String^>^ indexFields
)
```

JavaScript

```
function addUniqueIndexOn(indexName, ... indexFields);
```

Parameters

indexName
Type: System.
The name of the index

indexFields
Type: .[]
The list of fields of the index. Every field needs to implement IComparable.

See Also
IIndexManager Interface
NDatabase.Api Namespace

9.2.4.2.3 DeleteIndex Method

Delete existing index

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub DeleteIndex ( _
 indexName As String _
)
```

C#

```
void DeleteIndex(
 string indexName
)
```

Visual C++

```
void DeleteIndex(
 String^ indexName
)
```

JavaScript

```
function deleteIndex(indexName);
```

Parameters

indexName
Type: System.
Existing index name

See Also
IIndexManager Interface
NDatabase.Api Namespace

9.2.4.2.4 ExistIndex Method

To check if an index exist

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function ExistIndex ( _
 indexName As String _
) As Boolean
```

C#

```
bool ExistIndex(
 string indexName
)
```

Visual C++

```
bool ExistIndex(
 String^ indexName
)
```

JavaScript

```
function existIndex(indexName);
```

Parameters
indexName
Type: System.
Existing index name

See Also
IIndexManager Interface
NDatabase.Api Namespace

9.2.4.2.5 RebuildIndex Method

Rebuild existing index

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub RebuildIndex ( _
 indexName As String _
)
```

C#

```
void RebuildIndex(
```

```
 string indexName
)
```

Visual C++

```
void RebuildIndex(
 String^ indexName
)
```

JavaScript

```
function rebuildIndex(indexName);
```

Parameters

indexName

Type: System.
Existing index name

See Also

IIndexManager Interface
NDatabase.Api Namespace

9.2.5 IObjectRepresentation Interface

used to give the user an instance of an object representation, level2.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface IObjectRepresentation
```

C#

```
public interface IObjectRepresentation
```

Visual C++

```
public interface class IObjectRepresentation
```

JavaScript

```
NDatabase.Api.IObjectRepresentation = function();
NDatabase.Api.IObjectRepresentation.createInterface('NDatabase.Api.IObjectRepresentation');
```

Remarks

used to give the user an instance of an object representation, level2. The Object Representation encapsulates the NonNativeObjectInfo which is the internal object representation.

See Also

IObjectRepresentation Members
NDatabase.Api Namespace

9.2.5.1 IObjectRepresentation Members

The IObjectRepresentation type exposes the following members.

Methods

	Name	Description

© 2014 Jacek Spolnik

	GetObjectClassName	Retrieves the full object class name
	GetOid	Retrieves the oid of the object
	GetValueOf	Return the value of a specific attribute
	SetValueOf	Sets the value of a specific attribute

See Also
IObjectRepresentation Interface
NDatabase.Api Namespace

9.2.5.2 IObjectRepresentation Methods

The IObjectRepresentation type exposes the following members.

Methods

	Name	Description
	GetObjectClassName	Retrieves the full object class name
	GetOid	Retrieves the oid of the object
	GetValueOf	Return the value of a specific attribute
	SetValueOf	Sets the value of a specific attribute

See Also
IObjectRepresentation Interface
NDatabase.Api Namespace

9.2.5.2.1 GetObjectClassName Method

Retrieves the full object class name

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function GetObjectClassName As String
```

C#
```
string GetObjectClassName()
```

Visual C++
```
String^ GetObjectClassName()
```

JavaScript
```
function getObjectClassName();
```

See Also
IObjectRepresentation Interface
NDatabase.Api Namespace

9.2.5.2.2 GetOid Method

Retrieves the oid of the object

Namespace: NDatabase.Api

Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetOid As OID
```

C#

```
OID GetOid()
```

Visual C++

```
OID^ GetOid()
```

JavaScript

```
function getOid();
```

See Also
IObjectRepresentation Interface
NDatabase.Api Namespace

9.2.5.2.3 GetValueOf Method

Return the value of a specific attribute

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetValueOf ( _
 attributeName As String _
) As Object
```

C#

```
Object GetValueOf(
 string attributeName
)
```

Visual C++

```
Object^ GetValueOf(
 String^ attributeName
)
```

JavaScript

```
function getValueOf(attributeName);
```

Parameters
attributeName
Type: System.

See Also
IObjectRepresentation Interface
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.5.2.4 SetValueOf Method

Sets the value of a specific attribute

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub SetValueOf ( _
 attributeName As String, _
 value As Object _
)
```

C#

```
void SetValueOf(
 string attributeName,
 Object value
)
```

Visual C++

```
void SetValueOf(
 String^ attributeName,
 Object^ value
)
```

JavaScript

```
function setValueOf(attributeName, value);
```

Parameters

attributeName
Type: System.

value
Type: System.

See Also
IObjectRepresentation Interface
NDatabase.Api Namespace

9.2.6 IObjectSet(TItem) Interface

query resultset. The

Copy Code

```
ObjectSet
```

interface serves as a cursor to iterate through a set of objects retrieved by a query.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface IObjectSet(Of TItem) _
 Inherits ICollection(Of TItem), IEnumerable(Of T), _
 IEnumerable
```

C#

```
public interface IObjectSet<TItem> : ICollection<TItem>,
```

```
 IEnumerable<T>, IEnumerable
```

Visual C++

```
generic<typename TItem>
public interface class IObjectSet : ICollection<TItem>,
 IEnumerable<T>, IEnumerable
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Type Parameters
TItem

See Also
IObjectSet()
NDatabase.Api Namespace

9.2.6.1 IObjectSet(TItem) Members

The IObjectSet()

Methods

	Name	Description
	Add	(Inherited from ICollection(TItem)
	Clear	(Inherited from ICollection(TItem)
	Contains	(Inherited from ICollection(TItem)
	CopyTo	(Inherited from ICollection(TItem)
	GetEnumerator()	(Inherited from IEnumerable(T)
	GetEnumerator()	(Inherited from IEnumerable.)
	GetFirst	Return the first object of the collection, if exist

	HasNext	returns
		Copy Code
		`true`
		if the
		Copy Code
		`ObjectSet`
		has more elements.

	Next	returns the next object in the Copy Code `ObjectSet` .
	Remove	(Inherited from ICollection(TItem)

	Reset	resets the
		Copy Code
		`ObjectSet`
		cursor before the first element. A subsequent call to
		Copy Code
		`next()`

© 2014 Jacek Spolnik

		will return the first element.

Properties

	Name	Description
	Count	(Inherited from ICollection(TItem)
	IsReadOnly	(Inherited from ICollection(TItem)

See Also
IObjectSet()
NDatabase.Api Namespace

9.2.6.2 IObjectSet(TItem) Methods

The IObjectSet()

Methods

	Name	Description
	Add	(Inherited from ICollection(TItem)
	Clear	(Inherited from ICollection(TItem)
	Contains	(Inherited from ICollection(TItem)
	CopyTo	(Inherited from ICollection(TItem)
	GetEnumerator()	(Inherited from IEnumerable(T)
	GetEnumerator()	(Inherited from IEnumerable.)
	GetFirst	Return the first object of the collection, if exist

	HasNext	returns
		Copy Code
		`true`
		if the
		Copy Code
		`ObjectSet`
		has more elements.

	Next	returns the next object in the Copy Code `ObjectSet` .
	Remove	(Inherited from ICollection(TItem)

	Reset	resets the
		Copy Code
		`ObjectSet`
		cursor before the first element. A subsequent call to
		Copy Code
		`next()`
		will return the first element.

See Also
IObjectSet()

© 2014 Jacek Spolnik

NDatabase.Api Namespace

9.2.6.2.1 GetEnumerator Method

Overload List

	Name	Description
	GetEnumerator()	(Inherited from IEnumerable(T)
	GetEnumerator()	(Inherited from IEnumerable.)

See Also
IObjectSet()
IObjectSet()
NDatabase.Api Namespace

9.2.6.2.2 GetFirst Method

Return the first object of the collection, if exist

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function GetFirst As TItem
```

C#
```
TItem GetFirst()
```

Visual C++
```
TItem GetFirst()
```

JavaScript
```
function getFirst();
```

Return Value

See Also
IObjectSet()
NDatabase.Api Namespace

9.2.6.2.3 HasNext Method

returns

Copy Code
```
true
```

if the

Copy Code
```
ObjectSet
```

has more elements.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

© 2014 Jacek Spolnik

Visual Basic

```
Function HasNext As Boolean
```

C#

```
bool HasNext()
```

Visual C++

```
bool HasNext()
```

JavaScript

```
function hasNext();
```

Return Value

Copy Code

```
true
```

if the

Copy Code

```
ObjectSet
```

has more elements

See Also

IObjectSet()
NDatabase.Api Namespace

9.2.6.2.4 Next Method

returns the next object in the

Copy Code

```
ObjectSet
```

.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Next As TItem
```

C#

```
TItem Next()
```

Visual C++

```
TItem Next()
```

JavaScript

```
function next();
```

Return Value

the next object in the

Copy Code

```
ObjectSet
```

.

See Also
IObjectSet()
NDatabase.Api Namespace

9.2.6.2.5 Reset Method

resets the

```
ObjectSet
```

cursor before the first element. A subsequent call to

```
next()
```

will return the first element.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub Reset
```

C#

```
void Reset()
```

Visual C++

```
void Reset()
```

JavaScript

```
function reset();
```

See Also
IObjectSet()
NDatabase.Api Namespace

9.2.6.3 IObjectSet(TItem) Properties

The IObjectSet()

Properties

	Name	Description
	Count	(Inherited from ICollection(TItem)
	IsReadOnly	(Inherited from ICollection(TItem)

See Also
IObjectSet()
NDatabase.Api Namespace

9.2.7 IObjectValues Interface

Interface that will be implemented to hold a row of a result of an Object Values Query

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

© 2014 Jacek Spolnik

Syntax

Visual Basic

```
Public Interface IObjectValues
```

C#

```
public interface IObjectValues
```

Visual C++

```
public interface class IObjectValues
```

JavaScript

```
NDatabaseApiIObjectValues = function();
NDatabaseApiIObjectValues.createInterface('NDatabaseApiIObjectValues');
```

See Also
IObjectValues Members
NDatabase.Api Namespace

9.2.7.1 IObjectValues Members

The IObjectValues type exposes the following members.

Methods

	Name	Description
	GetByAlias	Get result by alias
	GetByIndex	Get result by index
	GetValues	Get values result

See Also
IObjectValues Interface
NDatabase.Api Namespace

9.2.7.2 IObjectValues Methods

The IObjectValues type exposes the following members.

Methods

	Name	Description
	GetByAlias	Get result by alias
	GetByIndex	Get result by index
	GetValues	Get values result

See Also
IObjectValues Interface
NDatabase.Api Namespace

9.2.7.2.1 GetByAlias Method

Get result by alias

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

© 2014 Jacek Spolnik

Syntax

Visual Basic

```
Function GetByAlias ( _
 alias As String _
) As Object
```

C#

```
Object GetByAlias(
 string alias
)
```

Visual C++

```
Object^ GetByAlias(
 String^ alias
)
```

JavaScript

```
function getByAlias(alias);
```

Parameters

alias

Type: System.
Alias for result

Return Value

Object result

See Also

IObjectValues Interface
NDatabase.Api Namespace

9.2.7.2.2 GetByIndex Method

Get result by index

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetByIndex ( _
 index As Integer _
) As Object
```

C#

```
Object GetByIndex(
 int index
)
```

Visual C++

```
Object^ GetByIndex(
 int index
)
```

JavaScript

© 2014 Jacek Spolnik

```
function getByIndex(index);
```

Parameters

index

Type: System.
Index for result

Return Value

Object result

See Also

IObjectValues Interface
NDatabase.Api Namespace

9.2.7.2.3 GetValues Method

Get values result

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetValues As Object()
```

C#

```
Object[] GetValues()
```

Visual C++

```
array<Object^>^ GetValues()
```

JavaScript

```
function getValues();
```

Return Value

Array of objects as values

See Also

IObjectValues Interface
NDatabase.Api Namespace

9.2.8 IOdb Interface

Database engine interface.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface IOdb _
 Inherits IDisposable
```

C#

```
public interface IOdb : IDisposable
```

© 2014 Jacek Spolnik

Visual C++

```
public interface class IOdb : IDisposable
```

JavaScript

```
NDatabase.Api.IOdb = function();
NDatabase.Api.IOdb.createInterface('NDatabase.Api.IOdb');
```

Remarks

The interface provides all methods to store, retrieve and delete objects and to change object state.

See Also

IOdb Members
NDatabase.Api Namespace

9.2.8.1 IOdb Members

The IOdb type exposes the following members.

Methods

	Name	Description
	AsQueryable(T)	As the queryable.
	Close	Closes the database.
	Commit	Commits all changes of the database.
	DefragmentTo	Defragments database to specified location.
	Delete(T)	Deletes the specified plain object.
	DeleteObjectWithId	Deletes the object with Object ID.
	Disconnect(T)	Disconnects the specified plain object from the current session.
	Dispose	(Inherited from IDisposable.)
	Ext	Get the extension of database interface to get the access to advanced functions
	GetObjectFromId	Gets the object from Object ID.
	GetObjectId(T)	Gets the object id of an database-aware object.
	GetRefactorManager	Gets the refactor manager.
	IndexManagerFor(T)	Get the indexes manager for specified object type.
	IsClosed	Determines whether the database is closed.
	Query(T)	Factory method to create a new instance of the query.
	QueryAndExecute(T)	Queries the database for instances of specified type and execute the query.
	Rollback	Rollbacks all uncommitted changes of the database.
	Store(T)	Stores the specified plain object.
	TriggerManagerFor(T)	Get the triggers manager for specified object type.
	ValuesQuery(T)()	Factory method to create a new instance of the values query.
	ValuesQuery(T)(OID)	Factory method to create a new instance of the values query for specified oid.

See Also

IOdb Interface
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.8.2 IOdb Methods

The IOdb type exposes the following members.

Methods

	Name	Description
	AsQueryable(T)	As the queryable.
	Close	Closes the database.
	Commit	Commits all changes of the database.
	DefragmentTo	Defragments database to specified location.
	Delete(T)	Deletes the specified plain object.
	DeleteObjectWithId	Deletes the object with Object ID.
	Disconnect(T)	Disconnects the specified plain object from the current session.
	Dispose	(Inherited from IDisposable.)
	Ext	Get the extension of database interface to get the access to advanced functions
	GetObjectFromId	Gets the object from Object ID.
	GetObjectId(T)	Gets the object id of an database-aware object.
	GetRefactorManager	Gets the refactor manager.
	IndexManagerFor(T)	Get the indexes manager for specified object type.
	IsClosed	Determines whether the database is closed.
	Query(T)	Factory method to create a new instance of the query.
	QueryAndExecute(T)	Queries the database for instances of specified type and execute the query.
	Rollback	Rollbacks all uncommitted changes of the database.
	Store(T)	Stores the specified plain object.
	TriggerManagerFor(T)	Get the triggers manager for specified object type.
	ValuesQuery(T)()	Factory method to create a new instance of the values query.
	ValuesQuery(T)(OID)	Factory method to create a new instance of the values query for specified oid.

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.1 AsQueryable(T) Method

As the queryable.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function AsQueryable(Of T) As ILinqQueryable(Of T)
```

C#

```
ILinqQueryable<T> AsQueryable<T>()
```

Visual C++

© 2014 Jacek Spolnik

```
generic<typename T>
ILinqQueryable<T>^ AsQueryable()
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Type Parameters

T

Plain object type.

Return Value

Queryable collection

Remarks

Interface for LINQ to NDatabase

See Also

IOdb Interface
NDatabase.Api Namespace

9.2.8.2.2 Close Method

Closes the database.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub Close
```

C#

```
void Close()
```

Visual C++

```
void Close()
```

JavaScript

```
function close();
```

Remarks

Automatically commit uncommitted changes of the database.

See Also

IOdb Interface
NDatabase.Api Namespace

9.2.8.2.3 Commit Method

Commits all changes of the database.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

© 2014 Jacek Spolnik

```
Sub Commit
```

C#

```
void Commit()
```

Visual C++

```
void Commit()
```

JavaScript

```
function commit();
```

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.4 DefragmentTo Method

Defragments database to specified location.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub DefragmentTo ( _
 newFileName As String _
)
```

C#

```
void DefragmentTo(
 string newFileName
)
```

Visual C++

```
void DefragmentTo(
 String^ newFileName
)
```

JavaScript

```
function defragmentTo(newFileName);
```

Parameters
newFileName
Type: System.
New name of the file.

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.5 Delete(T) Method

Deletes the specified plain object.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

© 2014 Jacek Spolnik

Syntax

Visual Basic

```
Function Delete(Of T As Class) ( _
 plainObject As T _
) As OID
```

C#

```
OID Delete<T>(
 T plainObject
)
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
OID^ Delete(
 T plainObject
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters

plainObject

Type: **T**
The plain object.

Type Parameters

T

Plain object type.

Return Value

Object ID of deleted plain object.

See Also

IOdb Interface
NDatabase.Api Namespace

9.2.8.2.6 DeleteObjectWithId Method

Deletes the object with Object ID.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub DeleteObjectWithId ( _
 oid As OID _
)
```

C#

```
void DeleteObjectWithId(
 OID oid
)
```

Visual C++

© 2014 Jacek Spolnik

```
void DeleteObjectWithId(
 OID^ oid
)
```

JavaScript

```
function deleteObjectWithId(oid);
```

Parameters

oid

Type: NDatabase.Api.
The oid of the object to be deleted.

See Also

IOdb Interface
NDatabase.Api Namespace

9.2.8.2.7 Disconnect(T) Method

Disconnects the specified plain object from the current session.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub Disconnect(Of T As Class) ( _
 plainObject As T _
)
```

C#

```
void Disconnect<T>(
 T plainObject
)
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
void Disconnect(
 T plainObject
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters

plainObject

Type: **T**
The plain object.

Type Parameters

T

Plain object type.

Remarks

The object is removed from the cache.

See Also

© 2014 Jacek Spolnik

IOdb Interface
NDatabase.Api Namespace

9.2.8.2.8 Ext Method

Get the extension of database interface to get the access to advanced functions

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Ext As IOdbExt
```

C#

```
IOdbExt Ext()
```

Visual C++

```
IOdbExt^ Ext()
```

JavaScript

```
function ext();
```

Return Value
Extended interface to database.

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.9 GetObjectFromId Method

Gets the object from Object ID.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetObjectFromId ( _
 id As OID _
) As Object
```

C#

```
Object GetObjectFromId(
 OID id
)
```

Visual C++

```
Object^ GetObjectFromId(
 OID^ id
)
```

JavaScript

```
function getObjectFromId(id);
```

Parameters

© 2014 Jacek Spolnik

id
Type: NDatabase.Api.
The Object ID.

Return Value
The object with the specified Object ID.

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.10 GetObjectId(T) Method

Gets the object id of an database-aware object.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetObjectId(Of T As Class) ( _
 plainObject As T _
) As OID
```

C#

```
OID GetObjectId<T>(
 T plainObject
)
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
OID^ GetObjectId(
 T plainObject
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters
plainObject
Type: **T**
The plain object.

Type Parameters
T
Plain object type.

Return Value
The database internal Object ID.

See Also
IOdb Interface
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.8.2.11 GetRefactorManager Method

Gets the refactor manager.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetRefactorManager As IRefactorManager
```

C#

```
IRefactorManager GetRefactorManager()
```

Visual C++

```
IRefactorManager^ GetRefactorManager()
```

JavaScript

```
function getRefactorManager();
```

Return Value
Refactor manager.

Remarks

Refactor manager allows on updating database schema, when classes definition were changed.

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.12 IndexManagerFor(T) Method

Get the indexes manager for specified object type.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function IndexManagerFor(Of T As Class) As IIndexManager
```

C#

```
IIndexManager IndexManagerFor<T>()
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
IIndexManager^ IndexManagerFor()
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Type Parameters
T
Plain object type.

© 2014 Jacek Spolnik

Return Value
Index manager.

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.13 IsClosed Method

Determines whether the database is closed.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function IsClosed As Boolean
```

C#
```
bool IsClosed()
```

Visual C++
```
bool IsClosed()
```

JavaScript
```
function isClosed();
```

Return Value
true if the database is closed; otherwise, false.

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.14 Query(T) Method

Factory method to create a new instance of the query.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function Query(Of T) As IQuery
```

C#
```
IQuery Query<T>()
```

Visual C++
```
generic<typename T>
IQuery^ Query()
```

JavaScript
```
JavaScript does not support generic types or methods.
```

Type Parameters

© 2014 Jacek Spolnik

T
Plain object type.

Return Value
New instance of query for the specified object type.

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.15 QueryAndExecute(T) Method

Queries the database for instances of specified type and execute the query.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function QueryAndExecute(Of T) As IObjectSet(Of T)
```

C#
```
IObjectSet<T> QueryAndExecute<T>()
```

Visual C++
```
generic<typename T>
IObjectSet<T>^ QueryAndExecute()
```

JavaScript
```
JavaScript does not support generic types or methods.
```

Type Parameters
T
Plain object type.

Return Value
List of stored objects that matches the query.

Remarks
Shortcut for

Copy Code
```
Query<T>().Execute<T>()
```

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.16 Rollback Method

Rollbacks all uncommitted changes of the database.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Sub Rollback
```

© 2014 Jacek Spolnik

C#

```
void Rollback()
```

Visual C++

```
void Rollback()
```

JavaScript

```
function rollback();
```

See Also
IOdb Interface
NDatabase.Api Namespace

9.2.8.2.17 Store(T) Method

Stores the specified plain object.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Store(Of T As Class) ( _
 plainObject As T _
) As OID
```

C#

```
OID Store<T>(
 T plainObject
)
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
OID^ Store(
 T plainObject
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters

plainObject
Type: **T**
The plain object.

Type Parameters

T
Plain object type.

Return Value
Object ID of stored plain object.

See Also
IOdb Interface
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.8.2.18 TriggerManagerFor(T) Method

Get the triggers manager for specified object type.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function TriggerManagerFor(Of T As Class) As ITriggerManager
```

C#

```
ITriggerManager TriggerManagerFor<T>()
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
ITriggerManager^ TriggerManagerFor()
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Type Parameters

T

Plain object type.

Return Value

Trigger manager.

See Also

IOdb Interface
NDatabase.Api Namespace

9.2.8.2.19 ValuesQuery Method

Overload List

	Name	Description
	ValuesQuery(T)()	Factory method to create a new instance of the values query.
	ValuesQuery(T)(OID)	Factory method to create a new instance of the values query for specified oid.

See Also

IOdb Interface
IOdb Members
NDatabase.Api Namespace

9.2.8.2.19.1 ValuesQuery(T) Method

Factory method to create a new instance of the values query.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

© 2014 Jacek Spolnik

```
Function ValuesQuery(Of T As Class) As IValuesQuery
```

C#

```
IValuesQuery ValuesQuery<T>()
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
IValuesQuery^ ValuesQuery()
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Type Parameters

T

Plain object type.

Return Value

New instance of values query for the specified object type.

See Also

IOdb Interface
ValuesQuery Overload
NDatabase.Api Namespace

9.2.8.2.19.2 ValuesQuery(T) Method (OID)

Factory method to create a new instance of the values query for specified oid.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function ValuesQuery(Of T As Class) ( _
 oid As OID _
) As IValuesQuery
```

C#

```
IValuesQuery ValuesQuery<T>(
 OID oid
)
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
IValuesQuery^ ValuesQuery(
 OID^ oid
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters

oid

Type: NDatabase.Api.
The oid of the stored plain object.

Type Parameters

T

Plain object type.

Return Value

New instance of values query for the specified object with a given oid.

See Also

IOdb Interface
ValuesQuery Overload
NDatabase.Api Namespace

9.2.9 IOdbComparable Interface

NDatabase wrapper to the native Comparable interface

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface IOdbComparable _
 Inherits IComparable
```

C#

```
public interface IOdbComparable : IComparable
```

Visual C++

```
public interface class IOdbComparable : IComparable
```

JavaScript

```
NDatabase.Api.IOdbComparable = function();
NDatabase.Api.IOdbComparable.createInterface('NDatabase.Api.IOdbComparable');
```

See Also

IOdbComparable Members
NDatabase.Api Namespace

9.2.9.1 IOdbComparable Members

The IOdbComparable type exposes the following members.

Methods

	Name	Description
	CompareTo	(Inherited from IComparable.)

See Also

IOdbComparable Interface
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.9.2 IOdbComparable Methods

The IOdbComparable type exposes the following members.

Methods

	Name	Description
	CompareTo	(Inherited from IComparable.)

See Also
IOdbComparable Interface
NDatabase.Api Namespace

9.2.10 IOdbExt Interface

An interface to provider extended access to ODB.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Public Interface IOdbExt
```

C#
```
public interface IOdbExt
```

Visual C++
```
public interface class IOdbExt
```

JavaScript
```
NDatabase.Api.IOdbExt = function();
NDatabase.Api.IOdbExt.createInterface('NDatabase.Api.IOdbExt');
```

See Also
IOdbExt Members
NDatabase.Api Namespace

9.2.10.1 IOdbExt Members

The IOdbExt type exposes the following members.

Methods

	Name	Description
	ConvertToExternalOID	Convert an OID to External OID
	GetDatabaseId	Get the Database ID
	GetDbId	Return the name of the database
	GetObjectCreationDate	Returns the object creation date in ms since 1/1/1970
	GetObjectExternalOID(T)	Gets the external OID of an Object.
	GetObjectUpdateDate	Returns the object last update date in ms since 1/1/1970
	GetObjectVersion	Returns the object version of the object that has the specified OID

© 2014 Jacek Spolnik

See Also
IOdbExt Interface
NDatabase.Api Namespace

9.2.10.2 IOdbExt Methods

The IOdbExt type exposes the following members.

Methods

	Name	Description
	ConvertToExternalOID	Convert an OID to External OID
	GetDatabaseId	Get the Database ID
	GetDbId	Return the name of the database
	GetObjectCreationDate	Returns the object creation date in ms since 1/1/1970
	GetObjectExternalOID(T)	Gets the external OID of an Object.
	GetObjectUpdateDate	Returns the object last update date in ms since 1/1/1970
	GetObjectVersion	Returns the object version of the object that has the specified OID

See Also
IOdbExt Interface
NDatabase.Api Namespace

9.2.10.2.1 ConvertToExternalOID Method

Convert an OID to External OID

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function ConvertToExternalOID ( _
 oid As OID _
) As IExternalOID
```

C#

```
IExternalOID ConvertToExternalOID(
 OID oid
)
```

Visual C++

```
IExternalOID^ ConvertToExternalOID(
 OID^ oid
)
```

JavaScript

```
function convertToExternalOID(oid);
```

Parameters

oid

Type: NDatabase.Api.

© 2014 Jacek Spolnik

Return Value
The external OID

See Also
IOdbExt Interface
NDatabase.Api Namespace

9.2.10.2.2 GetDatabaseId Method

Get the Database ID

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function GetDatabaseId As IDatabaseId
```

C#
```
IDatabaseId GetDatabaseId()
```

Visual C++
```
IDatabaseId^ GetDatabaseId()
```

JavaScript
```
function getDatabaseId();
```

See Also
IOdbExt Interface
NDatabase.Api Namespace

9.2.10.2.3 GetDbId Method

Return the name of the database

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function GetDbId As String
```

C#
```
string GetDbId()
```

Visual C++
```
String^ GetDbId()
```

JavaScript
```
function getDbId();
```

Return Value
the file name

See Also

IOdbExt Interface
NDatabase.Api Namespace

9.2.10.2.4 GetObjectCreationDate Method

Returns the object creation date in ms since 1/1/1970

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetObjectCreationDate ( _
 oid As OID _
) As Long
```

C#

```
long GetObjectCreationDate(
 OID oid
)
```

Visual C++

```
long long GetObjectCreationDate(
 OID^ oid
)
```

JavaScript

```
function getObjectCreationDate(oid);
```

Parameters

oid

Type: NDatabase.Api.

Return Value

The creation date

See Also

IOdbExt Interface
NDatabase.Api Namespace

9.2.10.2.5 GetObjectExternalOID(T) Method

Gets the external OID of an Object.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetObjectExternalOID(Of T As Class) ( _
 plainObject As T _
) As IExternalOID
```

C#

```
IExternalOID GetObjectExternalOID<T>(
 T plainObject
)
where T : class
```

© 2014 Jacek Spolnik

Visual C++

```
generic<typename T>
where T : refclass
IExternalOID^ GetObjectExternalOID(
 T plainObject
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters

plainObject

Type: **T**

Type Parameters

T

Remarks

Gets the external OID of an Object. The external OID contains the ID of the database + the oid of the object. The External OID can be used to identify objects outside the ODB database as it should be unique across databases. It can be used for example to implement a replication process.

See Also

IOdbExt Interface
NDatabase.Api Namespace

9.2.10.2.6 GetObjectUpdateDate Method

Returns the object last update date in ms since 1/1/1970

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetObjectUpdateDate ( _
 oid As OID _
) As Long
```

C#

```
long GetObjectUpdateDate(
 OID oid
)
```

Visual C++

```
long long GetObjectUpdateDate(
 OID^ oid
)
```

JavaScript

```
function getObjectUpdateDate(oid);
```

Parameters

oid

Type: NDatabase.Api.

Return Value
The last update date

See Also
IOdbExt Interface
NDatabase.Api Namespace

9.2.10.2.7 GetObjectVersion Method

Returns the object version of the object that has the specified OID

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetObjectVersion ( _
 oid As OID _
) As Integer
```

C#

```
int GetObjectVersion(
 OID oid
)
```

Visual C++

```
int GetObjectVersion(
 OID^ oid
)
```

JavaScript

```
function getObjectVersion(oid);
```

Parameters
oid
Type: NDatabase.Api.

See Also
IOdbExt Interface
NDatabase.Api Namespace

9.2.11 IOdbForTrigger Interface

Database engine interface (simplified for triggers purpose).

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface IOdbForTrigger _
 Inherits IDisposable
```

C#

```
public interface IOdbForTrigger : IDisposable
```

Visual C++

© 2014 Jacek Spolnik

```
public interface class IOdbForTrigger : IDisposable
```

JavaScript

```
NDatabase.Api.IOdbForTrigger = function();
NDatabase.Api.IOdbForTrigger.createInterface('NDatabase.Api.IOdbForTrigger');
```

Remarks

The interface provides all methods from IOdb which are allowed to access in triggers.

See Also

IOdbForTrigger Members
NDatabase.Api Namespace

9.2.11.1 IOdbForTrigger Members

The IOdbForTrigger type exposes the following members.

Methods

	Name	Description
	AsQueryable(T)	As the queryable.
	Delete(T)	Deletes the specified plain object.
	DeleteObjectWithId	Deletes the object with Object ID.
	Dispose	(Inherited from IDisposable.)
	Ext	Get the extension of database interface to get the access to advanced functions
	GetObjectFromId	Gets the object from Object ID.
	GetObjectId(T)	Gets the object id of an database-aware object.
	GetValues	Gets the values that matches the values query.
	IsClosed	Determines whether the database is closed.
	Query(T)	Factory method to create a new instance of the query.
	QueryAndExecute(T)	Queries the database for instances of specified type and execute the query.
	Store(T)	Stores the specified plain object.
	ValuesQuery(T)()	Factory method to create a new instance of the values query.
	ValuesQuery(T)(OID)	Factory method to create a new instance of the values query for specified oid.

See Also

IOdbForTrigger Interface
NDatabase.Api Namespace

9.2.11.2 IOdbForTrigger Methods

The IOdbForTrigger type exposes the following members.

Methods

	Name	Description
	AsQueryable(T)	As the queryable.
	Delete(T)	Deletes the specified plain object.
	DeleteObjectWithId	Deletes the object with Object ID.

© 2014 Jacek Spolnik

Dispose	(Inherited from IDisposable.)
Ext	Get the extension of database interface to get the access to advanced functions
GetObjectFromId	Gets the object from Object ID.
GetObjectId(T)	Gets the object id of an database-aware object.
GetValues	Gets the values that matches the values query.
IsClosed	Determines whether the database is closed.
Query(T)	Factory method to create a new instance of the query.
QueryAndExecute(T)	Queries the database for instances of specified type and execute the query.
Store(T)	Stores the specified plain object.
ValuesQuery(T)()	Factory method to create a new instance of the values query.
ValuesQuery(T)(OID)	Factory method to create a new instance of the values query for specified oid.

See Also
IOdbForTrigger Interface
NDatabase.Api Namespace

9.2.11.2.1 AsQueryable(T) Method

As the queryable.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function AsQueryable(Of T) As ILinqQueryable(Of T)
```

C#

```
ILinqQueryable<T> AsQueryable<T>()
```

Visual C++

```
generic<typename T>
ILinqQueryable<T>^ AsQueryable()
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Type Parameters

T

Plain object type.

Return Value
Queryable collection

Remarks

Interface for LINQ to NDatabase

See Also
IOdbForTrigger Interface
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.11.2.2 Delete(T) Method

Deletes the specified plain object.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Delete(Of T As Class) ( _
 plainObject As T _
) As OID
```

C#

```
OID Delete<T>(
 T plainObject
)
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
OID^ Delete(
 T plainObject
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters

plainObject
Type: **T**
The plain object.

Type Parameters

T
Plain object type.

Return Value
Object ID of deleted plain object.

See Also
IOdbForTrigger Interface
NDatabase.Api Namespace

9.2.11.2.3 DeleteObjectWithId Method

Deletes the object with Object ID.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub DeleteObjectWithId ( _
 oid As OID _
)
```

© 2014 Jacek Spolnik

C#

```
void DeleteObjectWithId(
 OID oid
)
```

Visual C++

```
void DeleteObjectWithId(
 OID^ oid
)
```

JavaScript

```
function deleteObjectWithId(oid);
```

Parameters

oid

Type: NDatabase.Api.
The oid of the object to be deleted.

See Also

IOdbForTrigger Interface
NDatabase.Api Namespace

9.2.11.2.4 Ext Method

Get the extension of database interface to get the access to advanced functions

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Ext As IOdbExt
```

C#

```
IOdbExt Ext()
```

Visual C++

```
IOdbExt^ Ext()
```

JavaScript

```
function ext();
```

Return Value

Extended interface to database.

See Also

IOdbForTrigger Interface
NDatabase.Api Namespace

9.2.11.2.5 GetObjectFromId Method

Gets the object from Object ID.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

© 2014 Jacek Spolnik

Visual Basic

```
Function GetObjectFromId ( _
 id As OID _
) As Object
```

C#

```
Object GetObjectFromId(
 OID id
)
```

Visual C++

```
Object^ GetObjectFromId(
 OID^ id
)
```

JavaScript

```
function getObjectFromId(id);
```

Parameters

id

Type: NDatabase.Api.
The Object ID.

Return Value

The object with the specified Object ID.

See Also

IOdbForTrigger Interface
NDatabase.Api Namespace

9.2.11.2.6 GetObjectId(T) Method

Gets the object id of an database-aware object.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetObjectId(Of T As Class) ( _
 plainObject As T _
) As OID
```

C#

```
OID GetObjectId<T>(
 T plainObject
)
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
OID^ GetObjectId(
 T plainObject
)
```

JavaScript

© 2014 Jacek Spolnik

```
JavaScript does not support generic types or methods.
```

Parameters

plainObject
Type: **T**
The plain object.

Type Parameters

T
Plain object type.

Return Value
The database internal Object ID.

See Also
IOdbForTrigger Interface
NDatabase.Api Namespace

9.2.11.2.7 GetValues Method

Gets the values that matches the values query.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetValues ( _
 query As IValuesQuery _
) As IValues
```

C#

```
IValues GetValues(
 IValuesQuery query
)
```

Visual C++

```
IValues^ GetValues(
 IValuesQuery^ query
)
```

JavaScript

```
function getValues(query);
```

Parameters

query
Type: NDatabase.Api.Query.
The values query.

Return Value
The list of values that matches the values query.

See Also
IOdbForTrigger Interface
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.11.2.8 IsClosed Method

Determines whether the database is closed.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function IsClosed As Boolean
```

C#

```
bool IsClosed()
```

Visual C++

```
bool IsClosed()
```

JavaScript

```
function isClosed();
```

Return Value
true if the database is closed; otherwise, false.

See Also
IOdbForTrigger Interface
NDatabase.Api Namespace

9.2.11.2.9 Query(T) Method

Factory method to create a new instance of the query.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Query(Of T) As IQuery
```

C#

```
IQuery Query<T>()
```

Visual C++

```
generic<typename T>
IQuery^ Query()
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Type Parameters
T
Plain object type.

Return Value
New instance of query for the specified object type.

See Also
IOdbForTrigger Interface

© 2014 Jacek Spolnik

NDatabase.Api Namespace

9.2.11.2.10 QueryAndExecute(T) Method

Queries the database for instances of specified type and execute the query.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function QueryAndExecute(Of T) As IObjectSet(Of T)
```

C#

```
IObjectSet<T> QueryAndExecute<T>()
```

Visual C++

```
generic<typename T>
IObjectSet<T>^ QueryAndExecute()
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Type Parameters

T

Plain object type.

Return Value

List of stored objects that matches the query.

Remarks

Shortcut for

Copy Code

```
Query<T>().Execute<T>()
```

See Also

IOdbForTrigger Interface
NDatabase.Api Namespace

9.2.11.2.11 Store(T) Method

Stores the specified plain object.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Store(Of T As Class) ( _
 plainObject As T _
) As OID
```

C#

```
OID Store<T>(
 T plainObject
)
```

© 2014 Jacek Spolnik

```
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
OID^ Store(
 T plainObject
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters

plainObject
Type: **T**
The plain object.

Type Parameters

T
Plain object type.

Return Value

Object ID of stored plain object.

See Also

IOdbForTrigger Interface
NDatabase.Api Namespace

9.2.11.2.12 ValuesQuery Method

Overload List

	Name	Description
	ValuesQuery(T)()	Factory method to create a new instance of the values query.
	ValuesQuery(T)(OID)	Factory method to create a new instance of the values query for specified oid.

See Also

IOdbForTrigger Interface
IOdbForTrigger Members
NDatabase.Api Namespace

9.2.11.2.12.1 ValuesQuery(T) Method

Factory method to create a new instance of the values query.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function ValuesQuery(Of T As Class) As IValuesQuery
```

C#

```
IValuesQuery ValuesQuery<T>()
where T : class
```

Visual C++

© 2014 Jacek Spolnik

```
generic<typename T>
where T : refclass
IValuesQuery^ ValuesQuery()
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Type Parameters

T

Plain object type.

Return Value

New instance of values query for the specified object type.

See Also

IOdbForTrigger Interface
ValuesQuery Overload
NDatabase.Api Namespace

9.2.11.2.12.2 ValuesQuery(T) Method (OID)

Factory method to create a new instance of the values query for specified oid.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function ValuesQuery(Of T As Class) ( _
 oid As OID _
) As IValuesQuery
```

C#

```
IValuesQuery ValuesQuery<T>(
 OID oid
)
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
IValuesQuery^ ValuesQuery(
 OID^ oid
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters

oid

Type: NDatabase.Api.
The oid of the stored plain object.

Type Parameters

T

Plain object type.

Return Value

© 2014 Jacek Spolnik

New instance of values query for the specified object with a given oid.

See Also
IOdbForTrigger Interface
ValuesQuery Overload
NDatabase.Api Namespace

9.2.12 IRefactorManager Interface

An interface for refactoring

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Public Interface IRefactorManager
```

C#
```
public interface IRefactorManager
```

Visual C++
```
public interface class IRefactorManager
```

JavaScript
```
NDatabase.Api.IRefactorManager = function();
NDatabase.Api.IRefactorManager.createInterface('NDatabase.Api.IRefactorManager');
```

See Also
IRefactorManager Members
NDatabase.Api Namespace

9.2.12.1 IRefactorManager Members

The IRefactorManager type exposes the following members.

Methods

	Name	Description
	AddField	Extend stored class by new field - refactoring
	RemoveField	Remove field from stored class - refactoring
	RenameClass	Rename stored class - refactoring
	RenameField	Rename field of stored class - refactoring

See Also
IRefactorManager Interface
NDatabase.Api Namespace

9.2.12.2 IRefactorManager Methods

The IRefactorManager type exposes the following members.

Methods

	Name	Description

© 2014 Jacek Spolnik

	AddField	Extend stored class by new field - refactoring
	RemoveField	Remove field from stored class - refactoring
	RenameClass	Rename stored class - refactoring
	RenameField	Rename field of stored class - refactoring

See Also
IRefactorManager Interface
NDatabase.Api Namespace

9.2.12.2.1 AddField Method

Extend stored class by new field - refactoring

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub AddField ( _
 type As Type, _
 fieldType As Type, _
 fieldName As String _
)
```

C#

```
void AddField(
 Type type,
 Type fieldType,
 string fieldName
)
```

Visual C++

```
void AddField(
 Type^ type,
 Type^ fieldType,
 String^ fieldName
)
```

JavaScript

```
function addField(type, fieldType, fieldName);
```

Parameters

type
Type: System.
Type of the class

fieldType
Type: System.
New field type

fieldName
Type: System.
New field name

See Also
IRefactorManager Interface
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.12.2.2 RemoveField Method

Remove field from stored class - refactoring

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub RemoveField ( _
 type As Type, _
 attributeName As String _
)
```

C#

```
void RemoveField(
 Type type,
 string attributeName
)
```

Visual C++

```
void RemoveField(
 Type^ type,
 String^ attributeName
)
```

JavaScript

```
function removeField(type, attributeName);
```

Parameters

type
Type: System.
Type of field to remove

attributeName
Type: System.
Name of field to remove

See Also
IRefactorManager Interface
NDatabase.Api Namespace

9.2.12.2.3 RenameClass Method

Rename stored class - refactoring

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub RenameClass ( _
 fullClassName As String, _
 newType As Type _
)
```

C#

```
void RenameClass(
 string fullClassName,
```

© 2014 Jacek Spolnik

```
 Type newType
)
```

Visual C++

```
void RenameClass(
 String^ fullClassName,
 Type^ newType
)
```

JavaScript

```
function renameClass(fullClassName, newType);
```

Parameters

fullClassName
Type: System.
Old class name

newType
Type: System.
New type to apply

See Also

IRefactorManager Interface
NDatabase.Api Namespace

9.2.12.2.4 RenameField Method

Rename field of stored class - refactoring

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub RenameField ( _
 type As Type, _
 attributeName As String, _
 newAttributeName As String _
)
```

C#

```
void RenameField(
 Type type,
 string attributeName,
 string newAttributeName
)
```

Visual C++

```
void RenameField(
 Type^ type,
 String^ attributeName,
 String^ newAttributeName
)
```

JavaScript

```
function renameField(type, attributeName, newAttributeName);
```

Parameters

type

Type: System.
Type of the class
attributeName
Type: System.
Old attribute name
newAttributeName
Type: System.
New attribute name

See Also
IRefactorManager Interface
NDatabase.Api Namespace

9.2.13 IValues Interface

The main interface of all Object Values query results of NDatabase ODB

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface IValues _
 Inherits IObjectSet(Of IObjectValues), ICollection(Of IObjectValues),  _
 IEnumerable(Of IObjectValues), IEnumerable
```

C#

```
public interface IValues : IObjectSet<IObjectValues>,
 ICollection<IObjectValues>, IEnumerable<IObjectValues>, IEnumerable
```

Visual C++

```
public interface class IValues : IObjectSet<IObjectValues^>,
 ICollection<IObjectValues^>, IEnumerable<IObjectValues^>, IEnumerable
```

JavaScript

```
NDatabase.Api.IValues = function();
NDatabase.Api.IValues.createInterface('NDatabase.Api.IValues');
```

See Also
IValues Members
NDatabase.Api Namespace

9.2.13.1 IValues Members

The IValues type exposes the following members.

Methods

	Name	Description
	Add	(Inherited from ICollection()
	Clear	(Inherited from ICollection()
	Contains	(Inherited from ICollection()
	CopyTo	(Inherited from ICollection()
	GetEnumerator()	(Inherited from IEnumerable()
	GetEnumerator()	(Inherited from IEnumerable.)
	GetFirst	Return the first object of the collection, if exist

© 2014 Jacek Spolnik

		(Inherited from IObjectSet()

	HasNext	returns
		Copy Code
		`true`
		if the
		Copy Code
		`ObjectSet`
		has more elements.
		(Inherited from IObjectSet()

	Next	returns the next object in the Copy Code `ObjectSet` . (Inherited from IObjectSet()
	NextValues	Get next values set
	Remove	(Inherited from ICollection()

	Reset	resets the
		Copy Code
		`ObjectSet`
		cursor before the first element. A subsequent call to
		Copy Code
		`next()`
		will return the first element.
		(Inherited from IObjectSet()

Properties

	Name	Description
	Count	(Inherited from ICollection()
	IsReadOnly	(Inherited from ICollection()

See Also

IValues Interface
NDatabase.Api Namespace

9.2.13.2 IValues Methods

The IValues type exposes the following members.

Methods

	Name	Description
	Add	(Inherited from ICollection()
	Clear	(Inherited from ICollection()
	Contains	(Inherited from ICollection()
	CopyTo	(Inherited from ICollection()
	GetEnumerator()	(Inherited from IEnumerable()
	GetEnumerator()	(Inherited from IEnumerable.)

© 2014 Jacek Spolnik

	GetFirst	Return the first object of the collection, if exist (Inherited from IObjectSet()

	HasNext	returns
		Copy Code
		`true`
		if the
		Copy Code
		`ObjectSet`
		has more elements.
		(Inherited from IObjectSet()

	Next	returns the next object in the Copy Code `ObjectSet` . (Inherited from IObjectSet()
	NextValues	Get next values set
	Remove	(Inherited from ICollection()

	Reset	resets the
		Copy Code
		`ObjectSet`
		cursor before the first element. A subsequent call to
		Copy Code
		`next()`
		will return the first element.
		(Inherited from IObjectSet()

See Also
IValues Interface
NDatabase.Api Namespace

9.2.13.2.1 GetEnumerator Method

Overload List

	Name	Description
	GetEnumerator()	(Inherited from IEnumerable()
	GetEnumerator()	(Inherited from IEnumerable.)

See Also
IValues Interface
IValues Members
NDatabase.Api Namespace

9.2.13.2.2 NextValues Method

Get next values set

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

© 2014 Jacek Spolnik

Visual Basic

```
Function NextValues As IObjectValues
```

C#

```
IObjectValues NextValues()
```

Visual C++

```
IObjectValues^ NextValues()
```

JavaScript

```
function nextValues();
```

Return Value
Next values

See Also
IValues Interface
NDatabase.Api Namespace

9.2.13.3 IValues Properties

The IValues type exposes the following members.

Properties

	Name	Description
	Count	(Inherited from ICollection()
	IsReadOnly	(Inherited from ICollection()

See Also
IValues Interface
NDatabase.Api Namespace

9.2.14 NonPersistentAttribute Class

Use when you don't want to serialize the field.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
<AttributeUsageAttribute(AttributeTargets.Field)> _
Public NotInheritable Class NonPersistentAttribute _
 Inherits Attribute
```

C#

```
[AttributeUsageAttribute(AttributeTargets.Field)]
public sealed class NonPersistentAttribute : Attribute
```

Visual C++

```
[AttributeUsageAttribute(AttributeTargets::Field)]
public ref class NonPersistentAttribute sealed : public Attribute
```

© 2014 Jacek Spolnik

```
JavaScript
NDatabase.Api.NonPersistentAttribute = function();
Type.createClass(
 'NDatabase.Api.NonPersistentAttribute',
 Attribute);
```

Remarks

In such case, mark the attribute with

```
Copy Code
[NonPersistent]
```

Inheritance Hierarchy

System.
 System.
  NDatabase.Api.

See Also

NonPersistentAttribute Members
NDatabase.Api Namespace

9.2.14.1 NonPersistentAttribute Members

The NonPersistentAttribute type exposes the following members.

Constructors

	Name	Description
	NonPersistentAttribute	

Methods

	Name	Description
	Equals	(Inherited from Attribute.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Attribute.)
	GetType	(Inherited from Object.)
	IsDefaultAttribute	(Inherited from Attribute.)
	Match	(Inherited from Attribute.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

Properties

	Name	Description
	TypeId	(Inherited from Attribute.)

Explicit Interface Implementations

	Name	Description

© 2014 Jacek Spolnik

	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)

See Also
NonPersistentAttribute Class
NDatabase.Api Namespace

9.2.14.2 NonPersistentAttribute Constructor

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Sub New
```

C#

```
public NonPersistentAttribute()
```

Visual C++

```
public:
NonPersistentAttribute()
```

JavaScript

```
NDatabase.Api.NonPersistentAttribute = function();
```

See Also
NonPersistentAttribute Class
NDatabase.Api Namespace

9.2.14.3 NonPersistentAttribute Methods

The NonPersistentAttribute type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Attribute.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Attribute.)
	GetType	(Inherited from Object.)
	IsDefaultAttribute	(Inherited from Attribute.)
	Match	(Inherited from Attribute.)
	MemberwiseClone	(Inherited from Object.)

© 2014 Jacek Spolnik

	ToString	(Inherited from Object.)

Explicit Interface Implementations

	Name	Description
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)

See Also
NonPersistentAttribute Class
NDatabase.Api Namespace

9.2.14.4 NonPersistentAttribute Properties

The NonPersistentAttribute type exposes the following members.

Properties

	Name	Description
	TypeId	(Inherited from Attribute.)

See Also
NonPersistentAttribute Class
NDatabase.Api Namespace

9.2.15 OdbConfiguration Class

The main NDatabase ODB Configuration class.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public NotInheritable Class OdbConfiguration
```

C#

```
public static class OdbConfiguration
```

Visual C++

```
public ref class OdbConfiguration abstract sealed
```

JavaScript

```
NDatabase.Api.OdbConfiguration = function();
Type.createClass(
 'NDatabase.Api.OdbConfiguration');
```

© 2014 Jacek Spolnik

Remarks

All engine configuration is done via this class.

Inheritance Hierarchy

System.
NDatabase.Api.

See Also

OdbConfiguration Members
NDatabase.Api Namespace

9.2.15.1 OdbConfiguration Members

The OdbConfiguration type exposes the following members.

Methods

	Name	Description
	DisableBTreeValidation	Disables the B tree validation.
	DisableLogging	Disables the logging.
	EnableBTreeValidation	Enables the B tree validation.
	EnableLogging	Enables the logging.
	GetIndexBTreeDegree	Get index BTree degree (on start it is equals to 20)
	IsBTreeValidationEnabled	Determines whether [is B tree validation enabled].
	IsLoggingEnabled	Determines whether [is logging enabled].
	RegisterLogger	Registers the logger.
	SetIndexBTreeDegree	Sets the index B tree degree.

Fields

	Name	Description
	DefaultIndexBTreeDegree	Default index BTree degree - 20

See Also

OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.2 OdbConfiguration Fields

The OdbConfiguration type exposes the following members.

Fields

	Name	Description
	DefaultIndexBTreeDegree	Default index BTree degree - 20

© 2014 Jacek Spolnik

See Also
OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.2.1 DefaultIndexBTreeDegree Field

Default index BTree degree - 20

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Public Shared ReadOnly DefaultIndexBTreeDegree As Integer
```

C#
```
public static readonly int DefaultIndexBTreeDegree
```

Visual C++
```
public:
static initonly int DefaultIndexBTreeDegree
```

JavaScript
```
NDatabase.Api.OdbConfiguration.defaultIndexBTreeDegree
```

See Also
OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.3 OdbConfiguration Methods

The OdbConfiguration type exposes the following members.

Methods

	Name	Description
	DisableBTreeValidation	Disables the B tree validation.
	DisableLogging	Disables the logging.
	EnableBTreeValidation	Enables the B tree validation.
	EnableLogging	Enables the logging.
	GetIndexBTreeDegree	Get index BTree degree (on start it is equals to 20)
	IsBTreeValidationEnabled	Determines whether [is B tree validation enabled].
	IsLoggingEnabled	Determines whether [is logging enabled].
	RegisterLogger	Registers the logger.
	SetIndexBTreeDegree	Sets the index B tree degree.

See Also
OdbConfiguration Class

© 2014 Jacek Spolnik

NDatabase.Api Namespace

9.2.15.3.1 DisableBTreeValidation Method

Disables the B tree validation.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Sub DisableBTreeValidation
```

C#

```
public static void DisableBTreeValidation()
```

Visual C++

```
public:
static void DisableBTreeValidation()
```

JavaScript

```
NDatabase.Api.OdbConfiguration.disableBTreeValidation = function();
```

See Also
OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.3.2 DisableLogging Method

Disables the logging.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Sub DisableLogging
```

C#

```
public static void DisableLogging()
```

Visual C++

```
public:
static void DisableLogging()
```

JavaScript

```
NDatabase.Api.OdbConfiguration.disableLogging = function();
```

See Also
OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.3.3 EnableBTreeValidation Method

Enables the B tree validation.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

© 2014 Jacek Spolnik

Syntax

Visual Basic

```
Public Shared Sub EnableBTreeValidation
```

C#

```
public static void EnableBTreeValidation()
```

Visual C++

```
public:
static void EnableBTreeValidation()
```

JavaScript

```
NDatabase.Api.OdbConfiguration.enableBTreeValidation = function();
```

Remarks

It is more safe to run with that (finding issues), but that hits performance.

See Also

OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.3.4 EnableLogging Method

Enables the logging.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Sub EnableLogging
```

C#

```
public static void EnableLogging()
```

Visual C++

```
public:
static void EnableLogging()
```

JavaScript

```
NDatabase.Api.OdbConfiguration.enableLogging = function();
```

See Also

OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.3.5 GetIndexBTreeDegree Method

Get index BTree degree (on start it is equals to 20)

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

© 2014 Jacek Spolnik

Visual Basic

```
Public Shared Function GetIndexBTreeDegree As Integer
```

C#

```
public static int GetIndexBTreeDegree()
```

Visual C++

```
public:
static int GetIndexBTreeDegree()
```

JavaScript

```
NDatabase.Api.OdbConfiguration.getIndexBTreeDegree = function();
```

Return Value
Degree of index BTree

Remarks

It is less safe to run without that (finding issues), but that improves performance.

See Also
OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.3.6 IsBTreeValidationEnabled Method

Determines whether [is B tree validation enabled].

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Function IsBTreeValidationEnabled As Boolean
```

C#

```
public static bool IsBTreeValidationEnabled()
```

Visual C++

```
public:
static bool IsBTreeValidationEnabled()
```

JavaScript

```
NDatabase.Api.OdbConfiguration.isBTreeValidationEnabled = function();
```

Return Value
`true` if [is B tree validation enabled]; otherwise, `false`.

See Also
OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.3.7 IsLoggingEnabled Method

Determines whether [is logging enabled].

Namespace: NDatabase.Api

© 2014 Jacek Spolnik

Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Function IsLoggingEnabled As Boolean
```

C#

```
public static bool IsLoggingEnabled()
```

Visual C++

```
public:
static bool IsLoggingEnabled()
```

JavaScript

```
NDatabase.Api.OdbConfiguration.isLoggingEnabled = function();
```

Return Value

true if [is logging enabled]; otherwise, false.

See Also

OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.3.8 RegisterLogger Method

Registers the logger.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Sub RegisterLogger ( _
 logger As ILogger _
)
```

C#

```
public static void RegisterLogger(
 ILogger logger
)
```

Visual C++

```
public:
static void RegisterLogger(
 ILogger^ logger
)
```

JavaScript

```
NDatabase.Api.OdbConfiguration.registerLogger = function(logger);
```

Parameters

logger

Type: NDatabase.
The logger.

Remarks

© 2014 Jacek Spolnik

Automatically enables the logging.

See Also
OdbConfiguration Class
NDatabase.Api Namespace

9.2.15.3.9 SetIndexBTreeDegree Method

Sets the index B tree degree.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared Sub SetIndexBTreeDegree ( _
 indexBTreeSize As Integer _
)
```

C#

```
public static void SetIndexBTreeDegree(
 int indexBTreeSize
)
```

Visual C++

```
public:
static void SetIndexBTreeDegree(
 int indexBTreeSize
)
```

JavaScript

```
NDatabase.Api.OdbConfiguration.setIndexBTreeDegree = function(indexBTreeSize);
```

Parameters

indexBTreeSize
Type: System.
Size of the index B tree.

Remarks

Default value is equal to 20.

See Also
OdbConfiguration Class
NDatabase.Api Namespace

9.2.16 OID Interface

Object ID interface

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface OID _
 Inherits IComparable(Of OID), IComparable
```

C#

```
public interface OID : IComparable<OID>,
 IComparable
```

Visual C++

```
public interface class OID : IComparable<OID^>,
 IComparable
```

JavaScript

```
NDatabaseApiOID = function();
NDatabaseApiOID.createInterface('NDatabaseApiOID');
```

See Also
OID Members
NDatabase.Api Namespace

9.2.16.1 OID Members

The OID type exposes the following members.

Methods

	Name	Description
	CompareTo(Object)	(Inherited from IComparable.)
	CompareTo(T)	(Inherited from IComparable()

Properties

	Name	Description
	ObjectId	Underlying long number - oid

See Also
OID Interface
NDatabase.Api Namespace

9.2.16.2 OID Methods

Methods

	Name	Description
	CompareTo(Object)	(Inherited from IComparable.)
	CompareTo(T)	(Inherited from IComparable()

See Also
OID Interface
NDatabase.Api Namespace

9.2.16.2.1 CompareTo Method

Overload List

	Name	Description
	CompareTo(Object)	(Inherited from IComparable.)

© 2014 Jacek Spolnik

	CompareTo(T)	(Inherited from IComparable()

See Also
OID Interface
OID Members
NDatabase.Api Namespace

9.2.16.3 OID Properties

The OID type exposes the following members.

Properties

	Name	Description
	ObjectId	Underlying long number - oid

See Also
OID Interface
NDatabase.Api Namespace

9.2.16.3.1 ObjectId Property

Underlying long number - oid

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
ReadOnly Property ObjectId As Long
 Get
```

C#

```
long ObjectId { get; }
```

Visual C++

```
property long long ObjectId {
 long long get ();
}
```

JavaScript

```
function get_objectId();
```

See Also
OID Interface
NDatabase.Api Namespace

9.2.17 OIDAttribute Class

Use when you want to enrich your class with OID. You can apply it on fields of type: long or OID.

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

© 2014 Jacek Spolnik

```
<AttributeUsageAttribute(AttributeTargets.Field)> _
Public NotInheritable Class OIDAttribute _
 Inherits Attribute
```

C#

```
[AttributeUsageAttribute(AttributeTargets.Field)]
public sealed class OIDAttribute : Attribute
```

Visual C++

```
[AttributeUsageAttribute(AttributeTargets::Field)]
public refclass OIDAttribute sealed : public Attribute
```

JavaScript

```
NDatabase.Api.OIDAttribute = function();
Type.createClass(
 'NDatabase.Api.OIDAttribute',
 Attribute);
```

Remarks

In such case, mark the attribute with

Copy Code

```
[OID]
```

Inheritance Hierarchy

System.
 System.
  NDatabase.Api.

See Also

OIDAttribute Members
NDatabase.Api Namespace

9.2.17.1 OIDAttribute Members

The OIDAttribute type exposes the following members.

Constructors

	Name	Description
	OIDAttribute	

Methods

	Name	Description
	Equals	(Inherited from Attribute.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Attribute.)
	GetType	(Inherited from Object.)
	IsDefaultAttribute	(Inherited from Attribute.)
	Match	(Inherited from Attribute.)
	MemberwiseClone	(Inherited from Object.)

© 2014 Jacek Spolnik

	ToString	(Inherited from Object.)

Properties

	Name	Description
	TypeId	(Inherited from Attribute.)

Explicit Interface Implementations

	Name	Description
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)

See Also
OIDAttribute Class
NDatabase.Api Namespace

9.2.17.2 OIDAttribute Constructor

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Sub New
```

C#

```
public OIDAttribute()
```

Visual C++

```
public:
OIDAttribute()
```

JavaScript

```
NDatabase.Api.OIDAttribute = function();
```

See Also
OIDAttribute Class
NDatabase.Api Namespace

9.2.17.3 OIDAttribute Methods

The OIDAttribute type exposes the following members.

Methods

© 2014 Jacek Spolnik

	Name	Description
	Equals	(Inherited from Attribute.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Attribute.)
	GetType	(Inherited from Object.)
	IsDefaultAttribute	(Inherited from Attribute.)
	Match	(Inherited from Attribute.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

Explicit Interface Implementations

	Name	Description
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)
	_Attribute.	(Inherited from Attribute.)

See Also
OIDAttribute Class
NDatabase.Api Namespace

9.2.17.4 OIDAttribute Properties

The OIDAttribute type exposes the following members.

Properties

	Name	Description
	TypeId	(Inherited from Attribute.)

See Also
OIDAttribute Class
NDatabase.Api Namespace

9.2.18 OrderByConstants Class

Constants used for ordering queries and creating ordered collection iterators

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public NotInheritable Class OrderByConstants
```

© 2014 Jacek Spolnik

C#

```
public sealed class OrderByConstants
```

Visual C++

```
public ref class OrderByConstants sealed
```

JavaScript

```
NDatabase.Api.OrderByConstants = function();
Type.createClass(
 'NDatabase.Api.OrderByConstants');
```

Inheritance Hierarchy

System.
NDatabase.Api.

See Also
OrderByConstants Members
NDatabase.Api Namespace

9.2.18.1 OrderByConstants Members

The OrderByConstants type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	IsOrderByAsc	Is ascending order
	IsOrderByDesc	Is descending order
	IsOrderByNone	Is no order set
	MemberwiseClone	(Inherited from Object.)
	ToString	(Overrides Object.()

Fields

	Name	Description
S	OrderByAsc	Ascending order
S	OrderByDesc	Descending order
S	OrderByNone	No order

See Also
OrderByConstants Class
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.2.18.2 OrderByConstants Fields

The OrderByConstants type exposes the following members.

Fields

	Name	Description
	OrderByAsc	Ascending order
	OrderByDesc	Descending order
	OrderByNone	No order

See Also
OrderByConstants Class
NDatabase.Api Namespace

9.2.18.2.1 OrderByAsc Field

Ascending order

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared ReadOnly OrderByAsc As OrderByConstants
```

C#

```
public static readonly OrderByConstants OrderByAsc
```

Visual C++

```
public:
static initonly OrderByConstants^ OrderByAsc
```

JavaScript

```
NDatabase.Api.OrderByConstants.orderByAsc
```

See Also
OrderByConstants Class
NDatabase.Api Namespace

9.2.18.2.2 OrderByDesc Field

Descending order

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared ReadOnly OrderByDesc As OrderByConstants
```

C#

```
public static readonly OrderByConstants OrderByDesc
```

© 2014 Jacek Spolnik

Visual C++

```
public:
static initonly OrderByConstants^ OrderByDesc
```

JavaScript

```
NDatabase.Api.OrderByConstants.orderByDesc
```

See Also
OrderByConstants Class
NDatabase.Api Namespace

9.2.18.2.3 OrderByNone Field

No order

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Shared ReadOnly OrderByNone As OrderByConstants
```

C#

```
public static readonly OrderByConstants OrderByNone
```

Visual C++

```
public:
static initonly OrderByConstants^ OrderByNone
```

JavaScript

```
NDatabase.Api.OrderByConstants.orderByNone
```

See Also
OrderByConstants Class
NDatabase.Api Namespace

9.2.18.3 OrderByConstants Methods

The OrderByConstants type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	IsOrderByAsc	Is ascending order
	IsOrderByDesc	Is descending order
	IsOrderByNone	Is no order set
	MemberwiseClone	(Inherited from Object.)
	ToString	(Overrides Object.()

© 2014 Jacek Spolnik

See Also
OrderByConstants Class
NDatabase.Api Namespace

9.2.18.3.1 IsOrderByAsc Method

Is ascending order

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Function IsOrderByAsc As Boolean
```

C#

```
public bool IsOrderByAsc()
```

Visual C++

```
public:
bool IsOrderByAsc()
```

JavaScript

```
function isOrderByAsc();
```

Return Value
True if ascending, false if not

See Also
OrderByConstants Class
NDatabase.Api Namespace

9.2.18.3.2 IsOrderByDesc Method

Is descending order

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Function IsOrderByDesc As Boolean
```

C#

```
public bool IsOrderByDesc()
```

Visual C++

```
public:
bool IsOrderByDesc()
```

JavaScript

```
function isOrderByDesc();
```

Return Value
True if descending, false if not

© 2014 Jacek Spolnik

See Also
OrderByConstants Class
NDatabase.Api Namespace

9.2.18.3.3 IsOrderByNone Method

Is no order set

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Function IsOrderByNone As Boolean
```

C#

```
public bool IsOrderByNone()
```

Visual C++

```
public:
bool IsOrderByNone()
```

JavaScript

```
function isOrderByNone();
```

Return Value
True if no order set, false if not

See Also
OrderByConstants Class
NDatabase.Api Namespace

9.2.18.3.4 ToString Method

Namespace: NDatabase.Api
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Overrides Function ToString As String
```

C#

```
public override string ToString()
```

Visual C++

```
public:
virtual String^ ToString() override
```

JavaScript

```
function toString();
```

See Also
OrderByConstants Class
NDatabase.Api Namespace

© 2014 Jacek Spolnik

9.3 NDatabase.Api.Query Namespace

Interfaces

	Interface	Description

	IConstraint	constraint to limit the objects returned upon query execution. Constraints are constructed by calling Query.Constrain(). Constraints can be joined with the methods And() and Or(). The methods to modify the constraint evaluation algorithm may be merged, to construct combined evaluation rules. Examples:
		• Copy Code
		`Constraint#Smaller().Equal()`
		for "smaller or equal"
		• Copy Code
		`Constraint#Like().Not()`
		for "not like"
		• Copy Code
		`Constraint#Greater().Equal().Not()`
		for "not greater or equal"

	ILinqQuery	NDatabase Linq Query interface
	ILinqQuery()	NDatabase Linq Query generic interface
	ILinqQueryable	IOrderedQueryable derived interface, Linq
	ILinqQueryable()	IOrderedQueryable<T> derived interface, Linq
	IQuery	handle to a node in the query graph. A node in the query graph can represent multiple classes, one class or an attribute of a class. The graph is automatically extended with attributes of added constraints (see constrain()) and upon calls to descend() that request nodes that do not yet exist. References to joined nodes in the query graph can be obtained by "walking" along the nodes of the graph with the method descend(). execute() evaluates the entire graph against all persistent objects. execute() can be called from any Query node of the graph. It will return an ObjectSet filled with objects of the class/classes that the node, it was called from, represents.
	IValuesQuery	Extending query with additional query metrics.

9.3.1 IConstraint Interface

constraint to limit the objects returned upon query execution. Constraints are constructed by calling Query.Constrain(). Constraints can be joined with the methods And() and Or(). The methods to modify the constraint evaluation algorithm may be merged, to construct combined evaluation rules. Examples:

• Copy Code

```
Constraint#Smaller().Equal()
```

for "smaller or equal"

• Copy Code

```
Constraint#Like().Not()
```

for "not like"

• Copy Code

```
Constraint#Greater().Equal().Not()
```

for "not greater or equal"

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

© 2014 Jacek Spolnik

Syntax

Visual Basic

```
Public Interface IConstraint
```

C#

```
public interface IConstraint
```

Visual C++

```
public interface class IConstraint
```

JavaScript

```
NDatabase.Api.Query.IConstraint = function();
NDatabase.Api.Query.IConstraint.createInterface('NDatabase.Api.Query.IConstraint');
```

See Also

IConstraint Members
NDatabase.Api.Query Namespace

9.3.1.1 IConstraint Members

The IConstraint type exposes the following members.

Methods

	Name	Description
	And	Links two IConstrains for AND evaluation.
	Contains	Sets the evaluation mode to containment comparison. Evaluation is dependant on the constrained query node: Copy Code `String` the persistent object is tested to contain a substring. arrays, collections the persistent object is tested to contain all elements of the constraining object.
	EndsWith	Sets evaluation to string ends with
	Equal	Sets the evaluation mode to Copy Code `==` .
	GetObject	returns the Object the query graph was constrained with to create this IConstraint.
	Greater	Sets the evaluation mode to Copy Code `>` .
	Identity	Sets the evaluation mode to identity comparison.
	InvariantLike	Sets the evaluation mode to "like" comparison. (Invariant mode)
	Like	Sets the evaluation mode to "like" comparison.
	Not	turns on not() comparison.
	Or	Links two IConstrains for OR evaluation.

© 2014 Jacek Spolnik

SizeEq	Check if size of the collection is equal
SizeGe	Check if size of the collection is greater or equal
SizeGt	Check if size of the collection is greater than
SizeLe	Check if size of the collection is less or equal
SizeLt	Check if size of the collection is less than
SizeNe	Check if size of the collection is not equal
Smaller	Sets the evaluation mode to Copy Code `<` .
StartsWith	Sets evaluation to string starts with

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2 IConstraint Methods

The IConstraint type exposes the following members.

Methods

	Name	Description
	And	Links two IConstrains for AND evaluation.
	Contains	Sets the evaluation mode to containment comparison. Evaluation is dependant on the constrained query node: Copy Code `String` the persistent object is tested to contain a substring. arrays, collections the persistent object is tested to contain all elements of the constraining object.
	EndsWith	Sets evaluation to string ends with
	Equal	Sets the evaluation mode to Copy Code `==` .
	GetObject	returns the Object the query graph was constrained with to create this IConstraint.
	Greater	Sets the evaluation mode to Copy Code `>` .
	Identity	Sets the evaluation mode to identity comparison.
	InvariantLike	Sets the evaluation mode to "like" comparison. (Invariant mode)
	Like	Sets the evaluation mode to "like" comparison.
	Not	turns on not() comparison.
	Or	Links two IConstrains for OR evaluation.

© 2014 Jacek Spolnik

	SizeEq	Check if size of the collection is equal
	SizeGe	Check if size of the collection is greater or equal
	SizeGt	Check if size of the collection is greater than
	SizeLe	Check if size of the collection is less or equal
	SizeLt	Check if size of the collection is less than
	SizeNe	Check if size of the collection is not equal
	Smaller	Sets the evaluation mode to Copy Code < .
	StartsWith	Sets evaluation to string starts with

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.1 And Method

Links two IConstrains for AND evaluation.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function And ( _
 with As IConstraint _
) As IConstraint
```

C#

```
IConstraint And(
 IConstraint with
)
```

Visual C++

```
IConstraint^ And(
 IConstraint^ with
)
```

JavaScript

```
function and(with);
```

Parameters

with
Type: NDatabase.Api.Query.
The other IConstraint

Return Value
A new IConstraint, that can be used for further calls to and() and or()

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.1.2.2 Contains Method

Sets the evaluation mode to containment comparison. Evaluation is dependant on the constrained query node:

	Copy Code
String	

the persistent object is tested to contain a substring.
arrays, collections
the persistent object is tested to contain all elements of the constraining object.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Contains As IConstraint
```

C#

```
IConstraint Contains()
```

Visual C++

```
IConstraint^ Contains()
```

JavaScript

```
function contains();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.3 EndsWith Method

Sets evaluation to string ends with

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function EndsWith ( _
 isCaseSensitive As Boolean _
) As IConstraint
```

C#

```
IConstraint EndsWith(
 bool isCaseSensitive
)
```

Visual C++

```
IConstraint^ EndsWith(
 bool isCaseSensitive
)
```

JavaScript

© 2014 Jacek Spolnik

```
function endsWith(isCaseSensitive);
```

Parameters

isCaseSensitive
Type: System.
Is case sensitive comparison

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.4 Equal Method

Sets the evaluation mode to

Copy Code

```
==
```

.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Equal As IConstraint
```

C#

```
IConstraint Equal()
```

Visual C++

```
IConstraint^ Equal()
```

JavaScript

```
function equal();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.5 GetObject Method

returns the Object the query graph was constrained with to create this IConstraint.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GetObject As Object
```

C#

© 2014 Jacek Spolnik

```
Object GetObject()
```

Visual C++

```
Object^ GetObject()
```

JavaScript

```
function getObject();
```

Return Value

The constraining object.

See Also

IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.6 Greater Method

Sets the evaluation mode to

	Copy Code

```
>
```

.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Greater As IConstraint
```

C#

```
IConstraint Greater()
```

Visual C++

```
IConstraint^ Greater()
```

JavaScript

```
function greater();
```

Return Value

this IConstraint to allow the chaining of method calls.

See Also

IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.7 Identity Method

Sets the evaluation mode to identity comparison.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Identity As IConstraint
```

C#

```
IConstraint Identity()
```

Visual C++

```
IConstraint^ Identity()
```

JavaScript

```
function identity();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.8 InvariantLike Method

Sets the evaluation mode to "like" comparison. (Invariant mode)

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function InvariantLike As IConstraint
```

C#

```
IConstraint InvariantLike()
```

Visual C++

```
IConstraint^ InvariantLike()
```

JavaScript

```
function invariantLike();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.9 Like Method

Sets the evaluation mode to "like" comparison.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Like As IConstraint
```

C#

```
IConstraint Like()
```

Visual C++

```
IConstraint^ Like()
```

JavaScript

```
function like();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.10 Not Method

turns on not() comparison.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Not As IConstraint
```

C#

```
IConstraint Not()
```

Visual C++

```
IConstraint^ Not()
```

JavaScript

```
function not();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.11 Or Method

Links two IConstrains for OR evaluation.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Or ( _
 with As IConstraint _
) As IConstraint
```

C#

© 2014 Jacek Spolnik

```
IConstraint Or(
 IConstraint with
)
```

Visual C++

```
IConstraint^ Or(
 IConstraint^ with
)
```

JavaScript

```
function or(with);
```

Parameters

with

Type: NDatabase.Api.Query.
The other IConstraint

Return Value

A new IConstraint, that can be used for further calls to and() and or()

See Also

IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.12 SizeEq Method

Check if size of the collection is equal

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function SizeEq As IConstraint
```

C#

```
IConstraint SizeEq()
```

Visual C++

```
IConstraint^ SizeEq()
```

JavaScript

```
function sizeEq();
```

Return Value

this IConstraint to allow the chaining of method calls.

See Also

IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.13 SizeGe Method

Check if size of the collection is greater or equal

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

© 2014 Jacek Spolnik

Syntax

Visual Basic

```
Function SizeGe As IConstraint
```

C#

```
IConstraint SizeGe()
```

Visual C++

```
IConstraint^ SizeGe()
```

JavaScript

```
function sizeGe();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.14 SizeGt Method

Check if size of the collection is greater than

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function SizeGt As IConstraint
```

C#

```
IConstraint SizeGt()
```

Visual C++

```
IConstraint^ SizeGt()
```

JavaScript

```
function sizeGt();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.15 SizeLe Method

Check if size of the collection is less or equal

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

© 2014 Jacek Spolnik

Visual Basic

```
Function SizeLe As IConstraint
```

C#

```
IConstraint SizeLe()
```

Visual C++

```
IConstraint^ SizeLe()
```

JavaScript

```
function sizeLe();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.16 SizeLt Method

Check if size of the collection is less than

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function SizeLt As IConstraint
```

C#

```
IConstraint SizeLt()
```

Visual C++

```
IConstraint^ SizeLt()
```

JavaScript

```
function sizeLt();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.17 SizeNe Method

Check if size of the collection is not equal

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

© 2014 Jacek Spolnik

```
Function SizeNe As IConstraint
```

C#

```
IConstraint SizeNe()
```

Visual C++

```
IConstraint^ SizeNe()
```

JavaScript

```
function sizeNe();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.18 Smaller Method

Sets the evaluation mode to

Copy Code

```
<
```

.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Smaller As IConstraint
```

C#

```
IConstraint Smaller()
```

Visual C++

```
IConstraint^ Smaller()
```

JavaScript

```
function smaller();
```

Return Value
this IConstraint to allow the chaining of method calls.

See Also
IConstraint Interface
NDatabase.Api.Query Namespace

9.3.1.2.19 StartsWith Method

Sets evaluation to string starts with

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

© 2014 Jacek Spolnik

Visual Basic

```
Function StartsWith ( _
 isCaseSensitive As Boolean _
) As IConstraint
```

C#

```
IConstraint StartsWith(
 bool isCaseSensitive
)
```

Visual C++

```
IConstraint^ StartsWith(
 bool isCaseSensitive
)
```

JavaScript

```
function startsWith(isCaseSensitive);
```

Parameters

isCaseSensitive
Type: System.
Is case sensitive comparison

Return Value

this IConstraint to allow the chaining of method calls.

See Also

IConstraint Interface
NDatabase.Api.Query Namespace

9.3.2 ILinqQuery Interface

NDatabase Linq Query interface

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface ILinqQuery _
 Inherits IEnumerable
```

C#

```
public interface ILinqQuery : IEnumerable
```

Visual C++

```
public interface class ILinqQuery : IEnumerable
```

JavaScript

```
NDatabase.Api.Query.ILinqQuery = function();
NDatabase.Api.Query.ILinqQuery.createInterface('NDatabase.Api.Query.ILinqQuery');
```

See Also

ILinqQuery Members
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.2.1 ILinqQuery Members

The ILinqQuery type exposes the following members.

Methods

	Name	Description
	GetEnumerator	(Inherited from IEnumerable.)

See Also
ILinqQuery Interface
NDatabase.Api.Query Namespace

9.3.2.2 ILinqQuery Methods

The ILinqQuery type exposes the following members.

Methods

	Name	Description
	GetEnumerator	(Inherited from IEnumerable.)

See Also
ILinqQuery Interface
NDatabase.Api.Query Namespace

9.3.3 ILinqQuery(T) Interface

NDatabase Linq Query generic interface

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface ILinqQuery(Of Out T) _
 Inherits ILinqQuery, IEnumerable(Of T), IEnumerable
```

C#

```
public interface ILinqQuery<out T> : ILinqQuery,
 IEnumerable<T>, IEnumerable
```

Visual C++

```
generic<typename T>
public interface class ILinqQuery : ILinqQuery,
 IEnumerable<T>, IEnumerable
```

JavaScript

JavaScript does not support generic types or methods.

Type Parameters
T

See Also
ILinqQuery()
NDatabase.Api.Query Namespace

9.3.3.1 ILinqQuery(T) Members

Methods

	Name	Description
	GetEnumerator()	(Inherited from IEnumerable(T)
	GetEnumerator()	(Inherited from IEnumerable.)

See Also
ILinqQuery()
NDatabase.Api.Query Namespace

9.3.3.2 ILinqQuery(T) Methods

Methods

	Name	Description
	GetEnumerator()	(Inherited from IEnumerable(T)
	GetEnumerator()	(Inherited from IEnumerable.)

See Also
ILinqQuery()
NDatabase.Api.Query Namespace

9.3.3.2.1 GetEnumerator Method

Overload List

	Name	Description
	GetEnumerator()	(Inherited from IEnumerable(T)
	GetEnumerator()	(Inherited from IEnumerable.)

See Also
ILinqQuery()
ILinqQuery()
NDatabase.Api.Query Namespace

9.3.4 ILinqQueryable Interface

IOrderedQueryable derived interface, Linq

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface ILinqQueryable _
 Inherits IOrderedQueryable, IQueryable, IEnumerable
```

C#

```
public interface ILinqQueryable : IOrderedQueryable,
 IQueryable, IEnumerable
```

Visual C++

© 2014 Jacek Spolnik

```
public interface class ILinqQueryable : IOrderedQueryable,
 IQueryable, IEnumerable
```

JavaScript

```
NDatabase.Api.Query.ILinqQueryable = function();
NDatabase.Api.Query.ILinqQueryable.createInterface('NDatabase.Api.Query.ILinqQueryable');
```

See Also
ILinqQueryable Members
NDatabase.Api.Query Namespace

9.3.4.1 ILinqQueryable Members

The ILinqQueryable type exposes the following members.

Methods

	Name	Description
	GetEnumerator	(Inherited from IEnumerable.)
	GetQuery	Get underliying linq query

Properties

	Name	Description
	ElementType	(Inherited from IQueryable.)
	Expression	(Inherited from IQueryable.)
	Provider	(Inherited from IQueryable.)

See Also
ILinqQueryable Interface
NDatabase.Api.Query Namespace

9.3.4.2 ILinqQueryable Methods

The ILinqQueryable type exposes the following members.

Methods

	Name	Description
	GetEnumerator	(Inherited from IEnumerable.)
	GetQuery	Get underliying linq query

See Also
ILinqQueryable Interface
NDatabase.Api.Query Namespace

9.3.4.2.1 GetQuery Method

Get underliying linq query

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

© 2014 Jacek Spolnik

Visual Basic

```
Function GetQuery As ILinqQuery
```

C#

```
ILinqQuery GetQuery()
```

Visual C++

```
ILinqQuery^ GetQuery()
```

JavaScript

```
function getQuery();
```

Return Value
Linq query

See Also
ILinqQueryable Interface
NDatabase.Api.Query Namespace

9.3.4.3 ILinqQueryable Properties

The ILinqQueryable type exposes the following members.

Properties

	Name	Description
	ElementType	(Inherited from IQueryable.)
	Expression	(Inherited from IQueryable.)
	Provider	(Inherited from IQueryable.)

See Also
ILinqQueryable Interface
NDatabase.Api.Query Namespace

9.3.5 ILinqQueryable(TElement) Interface

IOrderedQueryable<T> derived interface, Linq

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface ILinqQueryable(Of Out TElement) _
 Inherits IOrderedQueryable(Of TElement), IQueryable(Of TElement), _
 IEnumerable(Of T), ILinqQueryable, IOrderedQueryable, IQueryable, IEnumerable
```

C#

```
public interface ILinqQueryable<out TElement> : IOrderedQueryable<TElement>,
 IQueryable<TElement>, IEnumerable<T>, ILinqQueryable, IOrderedQueryable,
 IQueryable, IEnumerable
```

Visual C++

```
generic<typename TElement>
```

© 2014 Jacek Spolnik

```
public interface class ILinqQueryable : IOrderedQueryable<TElement>,
 IQueryable<TElement>, IEnumerable<T>, ILinqQueryable, IOrderedQueryable,
 IQueryable, IEnumerable
```

JavaScript

JavaScript does not support generic types or methods.

Type Parameters
TElement

See Also
ILinqQueryable()
NDatabase.Api.Query Namespace

9.3.5.1 ILinqQueryable(TElement) Members

The ILinqQueryable()

Methods

	Name	Description
	GetEnumerator()	(Inherited from IEnumerable(T)
	GetEnumerator()	(Inherited from IEnumerable.)
	GetQuery	Get underliying linq query (Inherited from ILinqQueryable.)

Properties

	Name	Description
	ElementType	(Inherited from IQueryable.)
	Expression	(Inherited from IQueryable.)
	Provider	(Inherited from IQueryable.)

See Also
ILinqQueryable()
NDatabase.Api.Query Namespace

9.3.5.2 ILinqQueryable(TElement) Methods

The ILinqQueryable()

Methods

	Name	Description
	GetEnumerator()	(Inherited from IEnumerable(T)
	GetEnumerator()	(Inherited from IEnumerable.)
	GetQuery	Get underliying linq query (Inherited from ILinqQueryable.)

See Also
ILinqQueryable()
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.5.2.1 GetEnumerator Method

Overload List

	Name	Description
	GetEnumerator()	(Inherited from IEnumerable(T)
	GetEnumerator()	(Inherited from IEnumerable.)

See Also
ILinqQueryable()
ILinqQueryable()
NDatabase.Api.Query Namespace

9.3.5.3 ILinqQueryable(TElement) Properties

The ILinqQueryable()

Properties

	Name	Description
	ElementType	(Inherited from IQueryable.)
	Expression	(Inherited from IQueryable.)
	Provider	(Inherited from IQueryable.)

See Also
ILinqQueryable()
NDatabase.Api.Query Namespace

9.3.6 IQuery Interface

handle to a node in the query graph. A node in the query graph can represent multiple classes, one class or an attribute of a class. The graph is automatically extended with attributes of added constraints (see constrain()) and upon calls to descend() that request nodes that do not yet exist. References to joined nodes in the query graph can be obtained by "walking" along the nodes of the graph with the method descend(). execute() evaluates the entire graph against all persistent objects. execute() can be called from any Query node of the graph. It will return an ObjectSet filled with objects of the class/classes that the node, it was called from, represents.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface IQuery
```

C#

```
public interface IQuery
```

Visual C++

```
public interface class IQuery
```

JavaScript

```
NDatabase.Api.Query.IQuery = function();
NDatabase.Api.Query.IQuery.createInterface('NDatabase.Api.Query.IQuery');
```

See Also

© 2014 Jacek Spolnik

IQuery Members
NDatabase.Api.Query Namespace

9.3.6.1 IQuery Members

The IQuery type exposes the following members.

Methods

	Name	Description
	Constrain	adds a constraint to this node. If the constraint contains attributes that are not yet present in the query graph, the query graph is extended accordingly.
	Count	Return count of filtered elements by defined query
	Descend	returns a reference to a descendant node in the query graph. If the node does not exist, it will be created. Query defined class represented in the query node is tested, whether it contains a field with the specified field name. The descendant Query node will be created from all possible candidates.
	Execute(T)()	executes the Query.
	Execute(T)(Boolean)	executes the Query.
	Execute(T)(Boolean, Int32, Int32)	executes the Query.
	OrderAscending	adds an ascending ordering criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called. @return this Query object to allow the chaining of method calls.
	OrderDescending	adds a descending order criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called. @return this Query object to allow the chaining of method calls.

See Also
IQuery Interface
NDatabase.Api.Query Namespace

9.3.6.2 IQuery Methods

The IQuery type exposes the following members.

Methods

	Name	Description
	Constrain	adds a constraint to this node. If the constraint contains attributes that are not yet present in the query graph, the query graph is extended accordingly.
	Count	Return count of filtered elements by defined query
	Descend	returns a reference to a descendant node in the query graph. If the node does not exist, it will be created. Query defined class represented in the query node is tested, whether it contains a field with the specified field name. The descendant Query node will be created from all possible candidates.
	Execute(T)()	executes the Query.
	Execute(T)(Boolean)	executes the Query.
	Execute(T)(Boolean, Int32, Int32)	executes the Query.
	OrderAscending	adds an ascending ordering criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called. @return this Query object to allow the chaining of method calls.
	OrderDescending	adds a descending order criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called. @return this Query object to allow the chaining of method calls.

© 2014 Jacek Spolnik

See Also
IQuery Interface
NDatabase.Api.Query Namespace

9.3.6.2.1 Constrain Method

adds a constraint to this node. If the constraint contains attributes that are not yet present in the query graph, the query graph is extended accordingly.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Constrain ( _
 value As Object _
) As IConstraint
```

C#

```
IConstraint Constrain(
 Object value
)
```

Visual C++

```
IConstraint^ Constrain(
 Object^ value
)
```

JavaScript

```
function constrain(value);
```

Parameters

value
Type: System.
constraint the constraint to be added to this Query.

Return Value
Constraint a new Constraint for this query node.

See Also
IQuery Interface
NDatabase.Api.Query Namespace

9.3.6.2.2 Count Method

Return count of filtered elements by defined query

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Count As Long
```

C#

```
long Count()
```

© 2014 Jacek Spolnik

Visual C++

```
long long Count()
```

JavaScript

```
function count();
```

Return Value
Number of filtered elements.

See Also
IQuery Interface
NDatabase.Api.Query Namespace

9.3.6.2.3 Descend Method

returns a reference to a descendant node in the query graph. If the node does not exist, it will be created. Query defined class represented in the query node is tested, whether it contains a field with the specified field name. The descendant Query node will be created from all possible candidates.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Descend ( _
 attributeName As String _
) As IQuery
```

C#

```
IQuery Descend(
 string attributeName
)
```

Visual C++

```
IQuery^ Descend(
 String^ attributeName
)
```

JavaScript

```
function descend(attributeName);
```

Parameters
attributeName
Type: System.
field path to the descendant.

Return Value
descendant Query node

See Also
IQuery Interface
NDatabase.Api.Query Namespace

9.3.6.2.4 Execute Method

Overload List

© 2014 Jacek Spolnik

	Name	Description
	Execute(T)()	executes the Query.
	Execute(T)(Boolean)	executes the Query.
	Execute(T)(Boolean, Int32, Int32)	executes the Query.

See Also
IQuery Interface
IQuery Members
NDatabase.Api.Query Namespace

9.3.6.2.4.1 Execute(T) Method

executes the Query.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function Execute(Of T) As IObjectSet(Of T)
```

C#
```
IObjectSet<T> Execute<T>()
```

Visual C++
```
generic<typename T>
IObjectSet<T>^ Execute()
```

JavaScript
```
JavaScript does not support generic types or methods.
```

Type Parameters
T

Return Value
ObjectSet - the result of the Query.

See Also
IQuery Interface
Execute Overload
NDatabase.Api.Query Namespace

9.3.6.2.4.2 Execute(T) Method (Boolean)

executes the Query.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function Execute(Of T As Class) ( _
 inMemory As Boolean _
) As IObjectSet(Of T)
```

© 2014 Jacek Spolnik

C#

```
IObjectSet<T> Execute<T>(
 bool inMemory
)
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
IObjectSet<T>^ Execute(
 bool inMemory
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters

inMemory

Type: System.
Indicates if all returned data should be loaded to memory (inMemory is true) or if the data should be lazy loaded (inMemory to false)

Type Parameters

T

Return Value

ObjectSet - the result of the Query.

See Also

IQuery Interface
Execute Overload
NDatabase.Api.Query Namespace

9.3.6.2.4.3 Execute(T) Method (Boolean, Int32, Int32)

executes the Query.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Execute(Of T As Class) ( _
 inMemory As Boolean, _
 startIndex As Integer, _
 endIndex As Integer _
) As IObjectSet(Of T)
```

C#

```
IObjectSet<T> Execute<T>(
 bool inMemory,
 int startIndex,
 int endIndex
)
where T : class
```

Visual C++

```
generic<typename T>
where T : refclass
IObjectSet<T>^ Execute(
 bool inMemory,
 int startIndex,
 int endIndex
)
```

JavaScript

```
JavaScript does not support generic types or methods.
```

Parameters

inMemory
Type: System.
Indicates if all returned data should be loaded to memory (inMemory is true) or if the data should be lazy loaded (inMemory to false)

startIndex
Type: System.
Start index for result page.

endIndex
Type: System.
End index for result page.

Type Parameters

T

Return Value

ObjectSet - the result of the Query.

See Also

IQuery Interface
Execute Overload
NDatabase.Api.Query Namespace

9.3.6.2.5 OrderAscending Method

adds an ascending ordering criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called. @return this Query object to allow the chaining of method calls.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function OrderAscending As IQuery
```

C#

```
IQuery OrderAscending()
```

Visual C++

```
IQuery^ OrderAscending()
```

JavaScript

```
function orderAscending();
```

See Also

IQuery Interface
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.6.2.6 OrderDescending Method

adds a descending order criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called. @return this Query object to allow the chaining of method calls.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function OrderDescending As IQuery
```

C#

```
IQuery OrderDescending()
```

Visual C++

```
IQuery^ OrderDescending()
```

JavaScript

```
function orderDescending();
```

See Also
IQuery Interface
NDatabase.Api.Query Namespace

9.3.7 IValuesQuery Interface

Extending query with additional query metrics.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface IValuesQuery _
 Inherits IQuery
```

C#

```
public interface IValuesQuery : IQuery
```

Visual C++

```
public interface class IValuesQuery : IQuery
```

JavaScript

```
NDatabase.Api.Query.IValuesQuery = function();
NDatabase.Api.Query.IValuesQuery.createInterface('NDatabase.Api.Query.IValuesQuery');
```

See Also
IValuesQuery Members
NDatabase.Api.Query Namespace

9.3.7.1 IValuesQuery Members

The IValuesQuery type exposes the following members.

Methods

© 2014 Jacek Spolnik

	Name	Description
	Avg(String)	Avgs the specified field name.
	Avg(String, String)	Averages the specified field name.
	Constrain	adds a constraint to this node. If the constraint contains attributes that are not yet present in the query graph, the query graph is extended accordingly. (Inherited from IQuery.)
	Count()	Return count of filtered elements by defined query (Inherited from IQuery.)
	Count(String)	Counts the objects that matches the specified values query.
	Descend	returns a reference to a descendant node in the query graph. If the node does not exist, it will be created. Query defined class represented in the query node is tested, whether it contains a field with the specified field name. The descendant Query node will be created from all possible candidates. (Inherited from IQuery.)
	Execute()	Gets the values that matches the values query.
	Execute(T)()	executes the Query. (Inherited from IQuery.)
	Execute(T)(Boolean)	executes the Query. (Inherited from IQuery.)
	Execute(T)(Boolean, Int32, Int32)	executes the Query. (Inherited from IQuery.)
	Field(String)	Field value for the specified field name.
	Field(String, String)	Field value for the specified field name.
	GroupBy	Groups by the specified field list.
	Max(String)	Max for the specified field name.
	Max(String, String)	Max for the specified field name.
	Min(String)	Min for the specified field name.
	Min(String, String)	Min for the specified field name.
	OrderAscending	adds an ascending ordering criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called. @return this Query object to allow the chaining of method calls. (Inherited from IQuery.)
	OrderDescending	adds a descending order criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called. @return this Query object to allow the chaining of method calls. (Inherited from IQuery.)
	SetReturnInstance	Enables or disables the return instance option.
	Size(String)	Size of the specified attribute name (collection or string).
	Size(String, String)	Size of the specified attribute name (collection or string).
	Sublist(String, Int32, Int32)	Sublists the specified attribute name (collection or string).
	Sublist(String, Int32, Int32, Boolean)	Sublists the specified attribute name (collection or string).
	Sublist(String, String, Int32, Int32)	Sublists the specified attribute name (collection or string).
	Sublist(String, String, Int32, Int32, Boolean)	Sublists the specified attribute name (collection or string).
	Sum(String)	Sums the specified field name.
	Sum(String, String)	Sums the specified field name.

See Also

© 2014 Jacek Spolnik

IValuesQuery Interface
NDatabase.Api.Query Namespace

9.3.7.2 IValuesQuery Methods

The IValuesQuery type exposes the following members.

Methods

	Name	Description
	Avg(String)	Avgs the specified field name.
	Avg(String, String)	Averages the specified field name.
	Constrain	adds a constraint to this node. If the constraint contains attributes that are not yet present in the query graph, the query graph is extended accordingly. (Inherited from IQuery.)
	Count()	Return count of filtered elements by defined query (Inherited from IQuery.)
	Count(String)	Counts the objects that matches the specified values query.
	Descend	returns a reference to a descendant node in the query graph. If the node does not exist, it will be created. Query defined class represented in the query node is tested, whether it contains a field with the specified field name. The descendant Query node will be created from all possible candidates. (Inherited from IQuery.)
	Execute()	Gets the values that matches the values query.
	Execute(T)()	executes the Query. (Inherited from IQuery.)
	Execute(T)(Boolean)	executes the Query. (Inherited from IQuery.)
	Execute(T)(Boolean, Int32, Int32)	executes the Query. (Inherited from IQuery.)
	Field(String)	Field value for the specified field name.
	Field(String, String)	Field value for the specified field name.
	GroupBy	Groups by the specified field list.
	Max(String)	Max for the specified field name.
	Max(String, String)	Max for the specified field name.
	Min(String)	Min for the specified field name.
	Min(String, String)	Min for the specified field name.
	OrderAscending	adds an ascending ordering criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called. @return this Query object to allow the chaining of method calls. (Inherited from IQuery.)
	OrderDescending	adds a descending order criteria to this node of the query graph. Multiple ordering criteria will be applied in the order they were called. @return this Query object to allow the chaining of method calls. (Inherited from IQuery.)
	SetReturnInstance	Enables or disables the return instance option.
	Size(String)	Size of the specified attribute name (collection or string).
	Size(String, String)	Size of the specified attribute name (collection or string).
	Sublist(String, Int32, Int32)	Sublists the specified attribute name (collection or string).
	Sublist(String, Int32, Int32, Boolean)	Sublists the specified attribute name (collection or string).
	Sublist(String, String,	Sublists the specified attribute name (collection or string).

© 2014 Jacek Spolnik

	Int32, Int32)	
	Sublist(String, String, Int32, Int32, Boolean)	Sublists the specified attribute name (collection or string).
	Sum(String)	Sums the specified field name.
	Sum(String, String)	Sums the specified field name.

See Also
IValuesQuery Interface
NDatabase.Api.Query Namespace

9.3.7.2.1 Avg Method

Overload List

	Name	Description
	Avg(String)	Avgs the specified field name.
	Avg(String, String)	Averages the specified field name.

See Also
IValuesQuery Interface
IValuesQuery Members
NDatabase.Api.Query Namespace

9.3.7.2.1.1 Avg Method (String)

Avgs the specified field name.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Avg ( _
 fieldName As String _
) As IValuesQuery
```

C#

```
IValuesQuery Avg(
 string fieldName
)
```

Visual C++

```
IValuesQuery^ Avg(
 String^ fieldName
)
```

JavaScript

```
function avg(fieldName);
```

Parameters

fieldName
Type: System.
Name of the field.

© 2014 Jacek Spolnik

Return Value
Values query with avg value.

See Also
IValuesQuery Interface
Avg Overload
NDatabase.Api.Query Namespace

9.3.7.2.1.2 Avg Method (String, String)

Averages the specified field name.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Avg ( _
 fieldName As String, _
 alias As String _
) As IValuesQuery
```

C#

```
IValuesQuery Avg(
 string fieldName,
 string alias
)
```

Visual C++

```
IValuesQuery^ Avg(
 String^ fieldName,
 String^ alias
)
```

JavaScript

```
function avg(fieldName, alias);
```

Parameters
fieldName
Type: System.
Name of the field.
alias
Type: System.
The alias for query value.

Return Value
Values query with alias itemset to avg value.

See Also
IValuesQuery Interface
Avg Overload
NDatabase.Api.Query Namespace

9.3.7.2.2 Count Method

Overload List

© 2014 Jacek Spolnik

	Name	Description
	Count()	Return count of filtered elements by defined query (Inherited from IQuery.)
	Count(String)	Counts the objects that matches the specified values query.

See Also
IValuesQuery Interface
IValuesQuery Members
NDatabase.Api.Query Namespace

9.3.7.2.2.1 Count Method (String)

Counts the objects that matches the specified values query.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Count ( _
 alias As String _
) As IValuesQuery
```

C#

```
IValuesQuery Count(
 string alias
)
```

Visual C++

```
IValuesQuery^ Count(
 String^ alias
)
```

JavaScript

```
function count(alias);
```

Parameters

alias
Type: System.
The alias for query value.

Return Value
Values query with alias itemset to count value.

See Also
IValuesQuery Interface
Count Overload
NDatabase.Api.Query Namespace

9.3.7.2.3 Execute Method

Overload List

	Name	Description
	Execute(T)()	executes the Query. (Inherited from IQuery.)

© 2014 Jacek Spolnik

	Execute()	Gets the values that matches the values query.
	Execute(T)(Boolean)	executes the Query. (Inherited from IQuery.)
	Execute(T)(Boolean, Int32, Int32)	executes the Query. (Inherited from IQuery.)

See Also
IValuesQuery Interface
IValuesQuery Members
NDatabase.Api.Query Namespace

9.3.7.2.3.1 Execute Method

Gets the values that matches the values query.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Function Execute As IValues
```

C#
```
IValues Execute()
```

Visual C++
```
IValues^ Execute()
```

JavaScript
```
function execute();
```

Return Value
The list of values that matches the values query.

See Also
IValuesQuery Interface
Execute Overload
NDatabase.Api.Query Namespace

9.3.7.2.4 Field Method

Overload List

	Name	Description
	Field(String)	Field value for the specified field name.
	Field(String, String)	Field value for the specified field name.

See Also
IValuesQuery Interface
IValuesQuery Members
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.7.2.4.1 Field Method (String)

Field value for the specified field name.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Field ( _
 fieldName As String _
) As IValuesQuery
```

C#

```
IValuesQuery Field(
 string fieldName
)
```

Visual C++

```
IValuesQuery^ Field(
 String^ fieldName
)
```

JavaScript

```
function field(fieldName);
```

Parameters

fieldName
Type: System.
Name of the field.

Return Value
Values query with field value.

See Also
IValuesQuery Interface
Field Overload
NDatabase.Api.Query Namespace

9.3.7.2.4.2 Field Method (String, String)

Field value for the specified field name.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Field ( _
 fieldName As String, _
 alias As String _
) As IValuesQuery
```

C#

```
IValuesQuery Field(
 string fieldName,
 string alias
)
```

© 2014 Jacek Spolnik

Visual C++

```
IValuesQuery^ Field(
 String^ fieldName,
 String^ alias
)
```

JavaScript

```
function field(fieldName, alias);
```

Parameters

fieldName
Type: System.
Name of the field.

alias
Type: System.
The alias for query value.

Return Value

Values query with alias itemset to field value.

See Also

IValuesQuery Interface
Field Overload
NDatabase.Api.Query Namespace

9.3.7.2.5 GroupBy Method

Groups by the specified field list.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function GroupBy ( _
 fieldList As String _
) As IValuesQuery
```

C#

```
IValuesQuery GroupBy(
 string fieldList
)
```

Visual C++

```
IValuesQuery^ GroupBy(
 String^ fieldList
)
```

JavaScript

```
function groupBy(fieldList);
```

Parameters

fieldList
Type: System.
The fields list.

Return Value

Values query with specified group by.

See Also
IValuesQuery Interface
NDatabase.Api.Query Namespace

9.3.7.2.6 Max Method

Overload List

	Name	Description
	Max(String)	Max for the specified field name.
	Max(String, String)	Max for the specified field name.

See Also
IValuesQuery Interface
IValuesQuery Members
NDatabase.Api.Query Namespace

9.3.7.2.6.1 Max Method (String)

Max for the specified field name.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Max ( _
 fieldName As String _
) As IValuesQuery
```

C#

```
IValuesQuery Max(
 string fieldName
)
```

Visual C++

```
IValuesQuery^ Max(
 String^ fieldName
)
```

JavaScript

```
function max(fieldName);
```

Parameters

fieldName
Type: System.
Name of the field.

Return Value
Values query max value.

See Also
IValuesQuery Interface
Max Overload
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.7.2.6.2 Max Method (String, String)

Max for the specified field name.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Max ( _
 fieldName As String, _
 alias As String _
) As IValuesQuery
```

C#

```
IValuesQuery Max(
 string fieldName,
 string alias
)
```

Visual C++

```
IValuesQuery^ Max(
 String^ fieldName,
 String^ alias
)
```

JavaScript

```
function max(fieldName, alias);
```

Parameters

fieldName
Type: System.
Name of the field.

alias
Type: System.
The alias for query value.

Return Value

Values query with alias itemset to max value.

See Also

IValuesQuery Interface
Max Overload
NDatabase.Api.Query Namespace

9.3.7.2.7 Min Method

Overload List

	Name	Description
	Min(String)	Min for the specified field name.
	Min(String, String)	Min for the specified field name.

See Also

IValuesQuery Interface
IValuesQuery Members
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.7.2.7.1 Min Method (String)

Min for the specified field name.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Min ( _
 fldName As String _
) As IValuesQuery
```

C#

```
IValuesQuery Min(
 string fldName
)
```

Visual C++

```
IValuesQuery^ Min(
 String^ fldName
)
```

JavaScript

```
function min(fldName);
```

Parameters

fieldName
Type: System.
Name of the field.

Return Value

Values query with min value.

See Also

IValuesQuery Interface
Min Overload
NDatabase.Api.Query Namespace

9.3.7.2.7.2 Min Method (String, String)

Min for the specified field name.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Min ( _
 fldName As String, _
 alias As String _
) As IValuesQuery
```

C#

```
IValuesQuery Min(
 string fldName,
 string alias
)
```

© 2014 Jacek Spolnik

Visual C++

```
IValuesQuery^ Min(
 String^ fieldName,
 String^ alias
)
```

JavaScript

```
function min(fieldName, alias);
```

Parameters

fieldName
Type: System.
Name of the field.

alias
Type: System.
The alias for query value.

Return Value

Values query with alias itemset to min value.

See Also

IValuesQuery Interface
Min Overload
NDatabase.Api.Query Namespace

9.3.7.2.8 SetReturnInstance Method

Enables or disables the return instance option.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub SetReturnInstance ( _
 returnInstance As Boolean _
)
```

C#

```
void SetReturnInstance(
 bool returnInstance
)
```

Visual C++

```
void SetReturnInstance(
 bool returnInstance
)
```

JavaScript

```
function setReturnInstance(returnInstance);
```

Parameters

returnInstance
Type: System.
if set to `true` [return instance].

Remarks

© 2014 Jacek Spolnik

To indicate if query execution must build instances or return object representation, Default value is true(return instance)

See Also
IValuesQuery Interface
NDatabase.Api.Query Namespace

9.3.7.2.9 Size Method

Overload List

	Name	Description
	Size(String)	Size of the specified attribute name (collection or string).
	Size(String, String)	Size of the specified attribute name (collection or string).

See Also
IValuesQuery Interface
IValuesQuery Members
NDatabase.Api.Query Namespace

9.3.7.2.9.1 Size Method (String)

Size of the specified attribute name (collection or string).

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Size ( _
 attributeName As String _
) As IValuesQuery
```

C#

```
IValuesQuery Size(
 string attributeName
)
```

Visual C++

```
IValuesQuery^ Size(
 String^ attributeName
)
```

JavaScript

```
function size(attributeName);
```

Parameters

attributeName
Type: System.
Name of the attribute (collection).

Return Value
Values query with size value.

See Also
IValuesQuery Interface
Size Overload
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.7.2.9.2 Size Method (String, String)

Size of the specified attribute name (collection or string).

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Size ( _
 attributeName As String, _
 alias As String _
) As IValuesQuery
```

C#

```
IValuesQuery Size(
 string attributeName,
 string alias
)
```

Visual C++

```
IValuesQuery^ Size(
 String^ attributeName,
 String^ alias
)
```

JavaScript

```
function size(attributeName, alias);
```

Parameters

attributeName
Type: System.
Name of the attribute (collection).

alias
Type: System.
The alias for query value.

Return Value

Values query with alias itemset to size value.

See Also

IValuesQuery Interface
Size Overload
NDatabase.Api.Query Namespace

9.3.7.2.10 Sublist Method

Overload List

	Name	Description
	Sublist(String, Int32, Int32)	Sublists the specified attribute name (collection or string).
	Sublist(String, Int32, Int32, Boolean)	Sublists the specified attribute name (collection or string).
	Sublist(String, String, Int32, Int32)	Sublists the specified attribute name (collection or string).

© 2014 Jacek Spolnik

	Sublist(String, String, Int32, Int32, Boolean)	Sublists the specified attribute name (collection or string).

See Also
IValuesQuery Interface
IValuesQuery Members
NDatabase.Api.Query Namespace

9.3.7.2.10.1 Sublist Method (String, Int32, Int32)

Sublists the specified attribute name (collection or string).

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Sublist ( _
 attributeName As String, _
 fromIndex As Integer, _
 toIndex As Integer _
) As IValuesQuery
```

C#

```
IValuesQuery Sublist(
 string attributeName,
 int fromIndex,
 int toIndex
)
```

Visual C++

```
IValuesQuery^ Sublist(
 String^ attributeName,
 int fromIndex,
 int toIndex
)
```

JavaScript

```
function sublist(attributeName, fromIndex, toIndex);
```

Parameters

attributeName
Type: System.
Name of the attribute (collection or string).

fromIndex
Type: System.
Start index.

toIndex
Type: System.
End index.

Return Value
Values query with sublist value.

See Also
IValuesQuery Interface
Sublist Overload
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.7.2.10.2 Sublist Method (String, Int32, Int32, Boolean)

Sublists the specified attribute name (collection or string).

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Sublist ( _
 attributeName As String, _
 fromIndex As Integer, _
 size As Integer, _
 throwException As Boolean _
) As IValuesQuery
```

C#

```
IValuesQuery Sublist(
 string attributeName,
 int fromIndex,
 int size,
 bool throwException
)
```

Visual C++

```
IValuesQuery^ Sublist(
 String^ attributeName,
 int fromIndex,
 int size,
 bool throwException
)
```

JavaScript

```
function sublist(attributeName, fromIndex, size, throwException);
```

Parameters

attributeName
Type: System.
Name of the attribute (collection or string).

fromIndex
Type: System.
Start index.

size
Type: System.
The size.

throwException
Type: System.
if set to `true` [throws exception].

Return Value

Values query with sublist value.

See Also

IValuesQuery Interface
Sublist Overload
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.7.2.10.3 Sublist Method (String, String, Int32, Int32)

Sublists the specified attribute name (collection or string).

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Sublist ( _
 attributeName As String, _
 alias As String, _
 fromIndex As Integer, _
 toIndex As Integer _
) As IValuesQuery
```

C#

```
IValuesQuery Sublist(
 string attributeName,
 string alias,
 int fromIndex,
 int toIndex
)
```

Visual C++

```
IValuesQuery^ Sublist(
 String^ attributeName,
 String^ alias,
 int fromIndex,
 int toIndex
)
```

JavaScript

```
function sublist(attributeName, alias, fromIndex, toIndex);
```

Parameters

attributeName
Type: System.
Name of the attribute (collection or string).

alias
Type: System.
The alias for query value.

fromIndex
Type: System.
Start index.

toIndex
Type: System.
End index.

Return Value

Values query with alias itemset to sublist value.

See Also

IValuesQuery Interface
Sublist Overload
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.7.2.10.4 Sublist Method (String, String, Int32, Int32, Boolean)

Sublists the specified attribute name (collection or string).

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Sublist ( _
 attributeName As String, _
 alias As String, _
 fromIndex As Integer, _
 size As Integer, _
 throwException As Boolean _
) As IValuesQuery
```

C#

```
IValuesQuery Sublist(
 string attributeName,
 string alias,
 int fromIndex,
 int size,
 bool throwException
)
```

Visual C++

```
IValuesQuery^ Sublist(
 String^ attributeName,
 String^ alias,
 int fromIndex,
 int size,
 bool throwException
)
```

JavaScript

```
function sublist(attributeName, alias, fromIndex, size, throwException);
```

Parameters

attributeName
Type: System.
Name of the attribute (collection or string).

alias
Type: System.
The alias for query value.

fromIndex
Type: System.
Start index.

size
Type: System.
The size.

throwException
Type: System.
if set to `true` [throws exception].

Return Value

Values query with alias itemset to sublist value.

See Also

IValuesQuery Interface

© 2014 Jacek Spolnik

Sublist Overload
NDatabase.Api.Query Namespace

9.3.7.2.11 Sum Method

Overload List

	Name	Description
	Sum(String)	Sums the specified field name.
	Sum(String, String)	Sums the specified field name.

See Also
IValuesQuery Interface
IValuesQuery Members
NDatabase.Api.Query Namespace

9.3.7.2.11.1 Sum Method (String)

Sums the specified field name.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Sum ( _
 fieldName As String _
) As IValuesQuery
```

C#

```
IValuesQuery Sum(
 string fieldName
)
```

Visual C++

```
IValuesQuery^ Sum(
 String^ fieldName
)
```

JavaScript

```
function sum(fieldName);
```

Parameters

fieldName
Type: System.
Name of the field.

Return Value
Values query with sum value.

See Also
IValuesQuery Interface
Sum Overload
NDatabase.Api.Query Namespace

© 2014 Jacek Spolnik

9.3.7.2.11.2 Sum Method (String, String)

Sums the specified field name.

Namespace: NDatabase.Api.Query
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Function Sum ( _
 fieldName As String, _
 alias As String _
) As IValuesQuery
```

C#

```
IValuesQuery Sum(
 string fieldName,
 string alias
)
```

Visual C++

```
IValuesQuery^ Sum(
 String^ fieldName,
 String^ alias
)
```

JavaScript

```
function sum(fieldName, alias);
```

Parameters

fieldName
Type: System.
Name of the field.

alias
Type: System.
The alias for query value.

Return Value

Values query with alias itemset to sum value.

See Also

IValuesQuery Interface
Sum Overload
NDatabase.Api.Query Namespace

9.4 NDatabase.Api.Triggers Namespace

Classes

	Class	Description
	DeleteTrigger	Abstract class - derive from it if you want to create delete trigger
	InsertTrigger	Abstract class - derive from it if you want to create insert trigger
	SelectTrigger	Abstract class - derive from it if you want to create select trigger
	Trigger	A simple base class for all triggers.

© 2014 Jacek Spolnik

	UpdateTrigger	Abstract class - derive from it if you want to create update trigger

Interfaces

	Interface	Description
	ITriggerManager	Triggers manager

9.4.1 DeleteTrigger Class

Abstract class - derive from it if you want to create delete trigger

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustInherit Class DeleteTrigger _
 Inherits Trigger
```

C#

```
public abstract class DeleteTrigger : Trigger
```

Visual C++

```
public ref class DeleteTrigger abstract : public Trigger
```

JavaScript

```
NDatabase.Api.Triggers.DeleteTrigger = function();
Type.createClass(
 'NDatabase.Api.Triggers.DeleteTrigger',
 NDatabase.Api.Triggers.Trigger);
```

Inheritance Hierarchy

System.
 NDatabase.Api.Triggers.
  NDatabase.Api.Triggers.

See Also

DeleteTrigger Members
NDatabase.Api.Triggers Namespace

9.4.1.1 DeleteTrigger Members

The DeleteTrigger type exposes the following members.

Constructors

	Name	Description
	DeleteTrigger	

Methods

	Name	Description
	AfterDelete	Action which will happen after delete

© 2014 Jacek Spolnik

	BeforeDelete	Action which will happen before delete
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

Properties

	Name	Description
	Odb	Access to NDatabase interface connected with created trigger (Inherited from Trigger.)

See Also
DeleteTrigger Class
NDatabase.Api.Triggers Namespace

9.4.1.2 DeleteTrigger Constructor

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic
```
Protected Sub New
```

C#
```
protected DeleteTrigger()
```

Visual C++
```
protected:
DeleteTrigger()
```

JavaScript
```
NDatabase.Api.Triggers.DeleteTrigger = function();
```

See Also
DeleteTrigger Class
NDatabase.Api.Triggers Namespace

9.4.1.3 DeleteTrigger Methods

The DeleteTrigger type exposes the following members.

Methods

	Name	Description
	AfterDelete	Action which will happen after delete
	BeforeDelete	Action which will happen before delete
	Equals	(Inherited from Object.)

© 2014 Jacek Spolnik

	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

See Also
DeleteTrigger Class
NDatabase.Api.Triggers Namespace

9.4.1.3.1 AfterDelete Method

Action which will happen after delete

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustOverride Sub AfterDelete ( _
 object As Object, _
 oid As OID _
)
```

C#

```
public abstract void AfterDelete(
 Object object,
 OID oid
)
```

Visual C++

```
public:
virtual void AfterDelete(
 Object^ object,
 OID^ oid
) abstract
```

JavaScript

```
function afterDelete(object, oid);
```

Parameters

object
Type: System.
Deleted object

oid
Type: NDatabase.Api.
Oid of deleted object

See Also
DeleteTrigger Class
NDatabase.Api.Triggers Namespace

© 2014 Jacek Spolnik

9.4.1.3.2 BeforeDelete Method

Action which will happen before delete

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustOverride Function BeforeDelete ( _
 object As Object, _
 oid As OID _
) As Boolean
```

C#

```
public abstract bool BeforeDelete(
 Object object,
 OID oid
)
```

Visual C++

```
public:
virtual bool BeforeDelete(
 Object^ object,
 OID^ oid
) abstract
```

JavaScript

```
function beforeDelete(object, oid);
```

Parameters

object
Type: System.
Deleted object

oid
Type: NDatabase.Api.
Oid of deleted object

Return Value
True if object was deleted, false if not

See Also
DeleteTrigger Class
NDatabase.Api.Triggers Namespace

9.4.1.4 DeleteTrigger Properties

The DeleteTrigger type exposes the following members.

Properties

	Name	Description
	Odb	Access to NDatabase interface connected with created trigger (Inherited from Trigger.)

See Also
DeleteTrigger Class
NDatabase.Api.Triggers Namespace

© 2014 Jacek Spolnik

9.4.2 InsertTrigger Class

Abstract class - derive from it if you want to create insert trigger

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustInherit Class InsertTrigger _
 Inherits Trigger
```

C#

```
public abstract class InsertTrigger : Trigger
```

Visual C++

```
public refclass InsertTrigger abstract : public Trigger
```

JavaScript

```
NDatabase.Api.Triggers.InsertTrigger = function();
Type.createClass(
 'NDatabase.Api.Triggers.InsertTrigger',
 NDatabase.Api.Triggers.Trigger);
```

Inheritance Hierarchy

System.
 NDatabase.Api.Triggers.
  NDatabase.Api.Triggers.

See Also
InsertTrigger Members
NDatabase.Api.Triggers Namespace

9.4.2.1 InsertTrigger Members

The InsertTrigger type exposes the following members.

Constructors

	Name	Description
	InsertTrigger	

Methods

	Name	Description
	AfterInsert	Action which will happen after insert
	BeforeInsert	Action which will happen before insert
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

© 2014 Jacek Spolnik

Properties

	Name	Description
	Odb	Access to NDatabase interface connected with created trigger (Inherited from Trigger.)

See Also
InsertTrigger Class
NDatabase.Api.Triggers Namespace

9.4.2.2 InsertTrigger Constructor

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Protected Sub New
```

C#

```
protected InsertTrigger()
```

Visual C++

```
protected:
InsertTrigger()
```

JavaScript

```
NDatabase.Api.Triggers.InsertTrigger = function();
```

See Also
InsertTrigger Class
NDatabase.Api.Triggers Namespace

9.4.2.3 InsertTrigger Methods

The InsertTrigger type exposes the following members.

Methods

	Name	Description
	AfterInsert	Action which will happen after insert
	BeforeInsert	Action which will happen before insert
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

See Also
InsertTrigger Class

© 2014 Jacek Spolnik

NDatabase.Api.Triggers Namespace

9.4.2.3.1 AfterInsert Method

Action which will happen after insert

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustOverride Sub AfterInsert ( _
 object As Object, _
 oid As OID _
)
```

C#

```
public abstract void AfterInsert(
 Object object,
 OID oid
)
```

Visual C++

```
public:
virtual void AfterInsert(
 Object^ object,
 OID^ oid
) abstract
```

JavaScript

```
function afterInsert(object, oid);
```

Parameters

object
Type: System.
Inserted object

oid
Type: NDatabase.Api.
Oid of inserted object

See Also
InsertTrigger Class
NDatabase.Api.Triggers Namespace

9.4.2.3.2 BeforeInsert Method

Action which will happen before insert

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustOverride Function BeforeInsert ( _
 object As Object _
) As Boolean
```

C#

© 2014 Jacek Spolnik

```
public abstract bool BeforeInsert(
 Object object
)
```

Visual C++

```
public:
virtual bool BeforeInsert(
 Object^ object
) abstract
```

JavaScript

```
function beforeInsert(object);
```

Parameters

object
Type: System.
Inserted object

Return Value

True if object inserted, false in other case

See Also

InsertTrigger Class
NDatabase.Api.Triggers Namespace

9.4.2.4 InsertTrigger Properties

The InsertTrigger type exposes the following members.

Properties

	Name	Description
	Odb	Access to NDatabase interface connected with created trigger (Inherited from Trigger.)

See Also

InsertTrigger Class
NDatabase.Api.Triggers Namespace

9.4.3 ITriggerManager Interface

Triggers manager

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Interface ITriggerManager
```

C#

```
public interface ITriggerManager
```

Visual C++

```
public interface class ITriggerManager
```

© 2014 Jacek Spolnik

JavaScript

```
NDatabase.Api.Triggers.ITriggerManager = function();
NDatabase.Api.Triggers.ITriggerManager.createInterface('NDatabase.Api.Triggers.ITriggerManager');
```

See Also
ITriggerManager Members
NDatabase.Api.Triggers Namespace

9.4.3.1 ITriggerManager Members

The ITriggerManager type exposes the following members.

Methods

	Name	Description
	AddDeleteTrigger	USed to add a delete trigger callback for the specific class
	AddInsertTrigger	Used to add an insert trigger callback for the specific class
	AddSelectTrigger	Used to add a select trigger callback for the specific class
	AddUpdateTrigger	Used to add an update trigger callback for the specific class

See Also
ITriggerManager Interface
NDatabase.Api.Triggers Namespace

9.4.3.2 ITriggerManager Methods

The ITriggerManager type exposes the following members.

Methods

	Name	Description
	AddDeleteTrigger	USed to add a delete trigger callback for the specific class
	AddInsertTrigger	Used to add an insert trigger callback for the specific class
	AddSelectTrigger	Used to add a select trigger callback for the specific class
	AddUpdateTrigger	Used to add an update trigger callback for the specific class

See Also
ITriggerManager Interface
NDatabase.Api.Triggers Namespace

9.4.3.2.1 AddDeleteTrigger Method

USed to add a delete trigger callback for the specific class

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub AddDeleteTrigger ( _
 trigger As DeleteTrigger _
)
```

C#

© 2014 Jacek Spolnik

```
void AddDeleteTrigger(
 DeleteTrigger trigger
)
```

Visual C++

```
void AddDeleteTrigger(
 DeleteTrigger^ trigger
)
```

JavaScript

```
function addDeleteTrigger(trigger);
```

Parameters

trigger

Type: NDatabase.Api.Triggers.

See Also

ITriggerManager Interface
NDatabase.Api.Triggers Namespace

9.4.3.2.2 AddInsertTrigger Method

Used to add an insert trigger callback for the specific class

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub AddInsertTrigger ( _
 trigger As InsertTrigger _
)
```

C#

```
void AddInsertTrigger(
 InsertTrigger trigger
)
```

Visual C++

```
void AddInsertTrigger(
 InsertTrigger^ trigger
)
```

JavaScript

```
function addInsertTrigger(trigger);
```

Parameters

trigger

Type: NDatabase.Api.Triggers.

See Also

ITriggerManager Interface
NDatabase.Api.Triggers Namespace

© 2014 Jacek Spolnik

9.4.3.2.3 AddSelectTrigger Method

Used to add a select trigger callback for the specific class

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub AddSelectTrigger ( _
 trigger As SelectTrigger _
)
```

C#

```
void AddSelectTrigger(
 SelectTrigger trigger
)
```

Visual C++

```
void AddSelectTrigger(
 SelectTrigger^ trigger
)
```

JavaScript

```
function addSelectTrigger(trigger);
```

Parameters

trigger

Type: NDatabase.Api.Triggers.

See Also

ITriggerManager Interface
NDatabase.Api.Triggers Namespace

9.4.3.2.4 AddUpdateTrigger Method

Used to add an update trigger callback for the specific class

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Sub AddUpdateTrigger ( _
 trigger As UpdateTrigger _
)
```

C#

```
void AddUpdateTrigger(
 UpdateTrigger trigger
)
```

Visual C++

```
void AddUpdateTrigger(
 UpdateTrigger^ trigger
)
```

© 2014 Jacek Spolnik

JavaScript

```
function addUpdateTrigger(trigger);
```

Parameters

trigger

Type: NDatabase.Api.Triggers.

See Also

ITriggerManager Interface
NDatabase.Api.Triggers Namespace

9.4.4 SelectTrigger Class

Abstract class - derive from it if you want to create select trigger

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustInherit Class SelectTrigger _
 Inherits Trigger
```

C#

```
public abstract class SelectTrigger : Trigger
```

Visual C++

```
public ref class SelectTrigger abstract : public Trigger
```

JavaScript

```
NDatabase.Api.Triggers.SelectTrigger = function();
Type.createClass(
 'NDatabase.Api.Triggers.SelectTrigger',
 NDatabase.Api.Triggers.Trigger);
```

Inheritance Hierarchy

System.
 NDatabase.Api.Triggers.
  NDatabase.Api.Triggers.

See Also

SelectTrigger Members
NDatabase.Api.Triggers Namespace

9.4.4.1 SelectTrigger Members

The SelectTrigger type exposes the following members.

Constructors

	Name	Description
	SelectTrigger	

Methods

© 2014 Jacek Spolnik

	Name	Description
	AfterSelect	Action which will happen after select
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

Properties

	Name	Description
	Odb	Access to NDatabase interface connected with created trigger (Inherited from Trigger.)

See Also
SelectTrigger Class
NDatabase.Api.Triggers Namespace

9.4.4.2 SelectTrigger Constructor

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Protected Sub New
```

C#

```
protected SelectTrigger()
```

Visual C++

```
protected:
SelectTrigger()
```

JavaScript

```
NDatabase.Api.Triggers.SelectTrigger = function();
```

See Also
SelectTrigger Class
NDatabase.Api.Triggers Namespace

9.4.4.3 SelectTrigger Methods

The SelectTrigger type exposes the following members.

Methods

	Name	Description
	AfterSelect	Action which will happen after select
	Equals	(Inherited from Object.)

© 2014 Jacek Spolnik

	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

See Also
SelectTrigger Class
NDatabase.Api.Triggers Namespace

9.4.4.3.1 AfterSelect Method

Action which will happen after select

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustOverride Sub AfterSelect ( _
 object As Object, _
 oid As OID _
)
```

C#

```
public abstract void AfterSelect(
 Object object,
 OID oid
)
```

Visual C++

```
public:
virtual void AfterSelect(
 Object^ object,
 OID^ oid
) abstract
```

JavaScript

```
function afterSelect(object, oid);
```

Parameters

object
Type: System.
Selected object

oid
Type: NDatabase.Api.
OID of selected object

See Also
SelectTrigger Class
NDatabase.Api.Triggers Namespace

© 2014 Jacek Spolnik

9.4.4.4 SelectTrigger Properties

The SelectTrigger type exposes the following members.

Properties

	Name	Description
	Odb	Access to NDatabase interface connected with created trigger (Inherited from Trigger.)

See Also
SelectTrigger Class
NDatabase.Api.Triggers Namespace

9.4.5 Trigger Class

A simple base class for all triggers.

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustInherit Class Trigger
```

C#

```
public abstract class Trigger
```

Visual C++

```
public refclass Trigger abstract
```

JavaScript

```
NDatabase.Api.Triggers.Trigger = function();
Type.createClass(
 'NDatabase.Api.Triggers.Trigger');
```

Inheritance Hierarchy

System.
NDatabase.Api.Triggers.
NDatabase.Api.Triggers.
NDatabase.Api.Triggers.
NDatabase.Api.Triggers.
NDatabase.Api.Triggers.

See Also
Trigger Members
NDatabase.Api.Triggers Namespace

9.4.5.1 Trigger Members

The Trigger type exposes the following members.

Constructors

	Name	Description
	Trigger	

© 2014 Jacek Spolnik

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

Properties

	Name	Description
	Odb	Access to NDatabase interface connected with created trigger

See Also
Trigger Class
NDatabase.Api.Triggers Namespace

9.4.5.2 Trigger Constructor

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Protected Sub New
```

C#

```
protected Trigger()
```

Visual C++

```
protected:
Trigger()
```

JavaScript

```
NDatabase.Api.Triggers.Trigger = function();
```

See Also
Trigger Class
NDatabase.Api.Triggers Namespace

9.4.5.3 Trigger Methods

The Trigger type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)

© 2014 Jacek Spolnik

	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

See Also
Trigger Class
NDatabase.Api.Triggers Namespace

9.4.5.4 Trigger Properties

The Trigger type exposes the following members.

Properties

	Name	Description
	Odb	Access to NDatabase interface connected with created trigger

See Also
Trigger Class
NDatabase.Api.Triggers Namespace

9.4.5.4.1 Odb Property

Access to NDatabase interface connected with created trigger

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Property Odb As IOdbForTrigger
 Get
 Friend Set
```

C#

```
public IOdbForTrigger Odb { get; internal set; }
```

Visual C++

```
public:
property IOdbForTrigger^ Odb {
 IOdbForTrigger^ get ();
 internal: void set (IOdbForTrigger^ value);
}
```

JavaScript

```
function get_odb();
function set_odb(value);
```

See Also
Trigger Class
NDatabase.Api.Triggers Namespace

© 2014 Jacek Spolnik

9.4.6 UpdateTrigger Class

Abstract class - derive from it if you want to create update trigger

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustInherit Class UpdateTrigger _
 Inherits Trigger
```

C#

```
public abstract class UpdateTrigger : Trigger
```

Visual C++

```
public refclass UpdateTrigger abstract : public Trigger
```

JavaScript

```
NDatabase.Api.Triggers.UpdateTrigger = function();
Type.createClass(
 'NDatabase.Api.Triggers.UpdateTrigger',
 NDatabase.Api.Triggers.Trigger);
```

Inheritance Hierarchy

System.
 NDatabase.Api.Triggers.
  NDatabase.Api.Triggers.

See Also
UpdateTrigger Members
NDatabase.Api.Triggers Namespace

9.4.6.1 UpdateTrigger Members

The UpdateTrigger type exposes the following members.

Constructors

	Name	Description
	UpdateTrigger	

Methods

	Name	Description
	AfterUpdate	Action which will happen after update
	BeforeUpdate	Action which will happen before update
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

© 2014 Jacek Spolnik

Properties

	Name	Description
	Odb	Access to NDatabase interface connected with created trigger (Inherited from Trigger.)

See Also
UpdateTrigger Class
NDatabase.Api.Triggers Namespace

9.4.6.2 UpdateTrigger Constructor

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Protected Sub New
```

C#

```
protected UpdateTrigger()
```

Visual C++

```
protected:
UpdateTrigger()
```

JavaScript

```
NDatabase.Api.Triggers.UpdateTrigger = function();
```

See Also
UpdateTrigger Class
NDatabase.Api.Triggers Namespace

9.4.6.3 UpdateTrigger Methods

The UpdateTrigger type exposes the following members.

Methods

	Name	Description
	AfterUpdate	Action which will happen after update
	BeforeUpdate	Action which will happen before update
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetHashCode	(Inherited from Object.)
	GetType	(Inherited from Object.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Object.)

See Also
UpdateTrigger Class

© 2014 Jacek Spolnik

NDatabase.Api.Triggers Namespace

9.4.6.3.1 AfterUpdate Method

Action which will happen after update

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustOverride Sub AfterUpdate ( _
 oldObjectRepresentation As IObjectRepresentation, _
 newObject As Object, _
 oid As OID _
)
```

C#

```
public abstract void AfterUpdate(
 IObjectRepresentation oldObjectRepresentation,
 Object newObject,
 OID oid
)
```

Visual C++

```
public:
virtual void AfterUpdate(
 IObjectRepresentation^ oldObjectRepresentation,
 Object^ newObject,
 OID^ oid
) abstract
```

JavaScript

```
function afterUpdate(oldObjectRepresentation, newObject, oid);
```

Parameters

oldObjectRepresentation
Type: NDatabase.Api.
Object representation

newObject
Type: System.
Updated object

oid
Type: NDatabase.Api.
Oid of updated object

See Also

UpdateTrigger Class
NDatabase.Api.Triggers Namespace

9.4.6.3.2 BeforeUpdate Method

Action which will happen before update

Namespace: NDatabase.Api.Triggers
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public MustOverride Function BeforeUpdate ( _
 oldObjectRepresentation As IObjectRepresentation, _
 newObject As Object, _
 oid As OID _
) As Boolean
```

C#

```
public abstract bool BeforeUpdate(
 IObjectRepresentation oldObjectRepresentation,
 Object newObject,
 OID oid
)
```

Visual C++

```
public:
virtual bool BeforeUpdate(
 IObjectRepresentation^ oldObjectRepresentation,
 Object^ newObject,
 OID^ oid
) abstract
```

JavaScript

```
function beforeUpdate(oldObjectRepresentation, newObject, oid);
```

Parameters

oldObjectRepresentation
Type: NDatabase.Api.
Object representation

newObject
Type: System.
Updated object

oid
Type: NDatabase.Api.
Oid of updated object

Return Value

True if updated, in other case false

See Also

UpdateTrigger Class
NDatabase.Api.Triggers Namespace

9.4.6.4 UpdateTrigger Properties

The UpdateTrigger type exposes the following members.

Properties

	Name	Description
	Odb	Access to NDatabase interface connected with created trigger (Inherited from Trigger.)

See Also

UpdateTrigger Class
NDatabase.Api.Triggers Namespace

© 2014 Jacek Spolnik

9.5 NDatabase.Exceptions Namespace

Classes

	Class	Description
	BTreeException	Exception raised when error in BTrees will appear
	BTreeNodeValidationException	Exception raised when error in BTrees will appear (validation error)
	CorruptedDatabaseException	An exception thrown by ODB when a corrupted block is found
	DuplicatedKeyException	Exception raised when error in BTrees will appear (Duplicated key)
	LinqQueryException	NDatabase exception raised during processing linq query
	OdbRuntimeException	Generic ODB Runtime exception : Used to report all problems.

9.5.1 BTreeException Class

Exception raised when error in BTrees will appear

Namespace: NDatabase.Exceptions
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Class BTreeException _
 Inherits OdbRuntimeException
```

C#

```
public class BTreeException : OdbRuntimeException
```

Visual C++

```
public ref class BTreeException : public OdbRuntimeException
```

JavaScript

```
NDatabase.Exceptions.BTreeException = function();
Type.createClass(
 'NDatabase.Exceptions.BTreeException',
 NDatabase.Exceptions.OdbRuntimeException);
```

Inheritance Hierarchy

System.
 System.
  NDatabase.Exceptions.
   NDatabase.Exceptions.
    NDatabase.Exceptions.

See Also

BTreeException Members
NDatabase.Exceptions Namespace

© 2014 Jacek Spolnik

9.5.1.1 BTreeException Members

The BTreeException type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)
	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also

BTreeException Class
NDatabase.Exceptions Namespace

9.5.1.2 BTreeException Methods

The BTreeException type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)

© 2014 Jacek Spolnik

	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

See Also
BTreeException Class
NDatabase.Exceptions Namespace

9.5.1.3 BTreeException Properties

The BTreeException type exposes the following members.

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)
	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also
BTreeException Class
NDatabase.Exceptions Namespace

9.5.2 BTreeNodeValidationException Class

Exception raised when error in BTrees will appear (validation error)

Namespace: NDatabase.Exceptions
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public NotInheritable Class BTreeNodeValidationException _
 Inherits OdbRuntimeException
```

C#

```
public sealed class BTreeNodeValidationException : OdbRuntimeException
```

Visual C++

```
public ref class BTreeNodeValidationException sealed : public OdbRuntimeException
```

JavaScript

```
NDatabase.Exceptions.BTreeNodeValidationException = function();
Type.createClass(
 'NDatabase.Exceptions.BTreeNodeValidationException',
 NDatabase.Exceptions.OdbRuntimeException);
```

© 2014 Jacek Spolnik

Inheritance Hierarchy

System.
System.
NDatabase.Exceptions.
NDatabase.Exceptions.

See Also
BTreeNodeValidationException Members
NDatabase.Exceptions Namespace

9.5.2.1 BTreeNodeValidationException Members

The BTreeNodeValidationException type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)
	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also
BTreeNodeValidationException Class
NDatabase.Exceptions Namespace

9.5.2.2 BTreeNodeValidationException Methods

The BTreeNodeValidationException type exposes the following members.

Methods

© 2014 Jacek Spolnik

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

See Also
BTreeNodeValidationException Class
NDatabase.Exceptions Namespace

9.5.2.3 BTreeNodeValidationException Properties

The BTreeNodeValidationException type exposes the following members.

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)
	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also
BTreeNodeValidationException Class
NDatabase.Exceptions Namespace

9.5.3 CorruptedDatabaseException Class

An exception thrown by ODB when a corrupted block is found

Namespace: NDatabase.Exceptions
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public NotInheritable Class CorruptedDatabaseException _
 Inherits OdbRuntimeException
```

C#

```
public sealed class CorruptedDatabaseException : OdbRuntimeException
```

© 2014 Jacek Spolnik

Visual C++

```
public ref class CorruptedDatabaseException sealed : public OdbRuntimeException
```

JavaScript

```
NDatabase.Exceptions.CorruptedDatabaseException = function();
Type.createClass(
 'NDatabase.Exceptions.CorruptedDatabaseException',
 NDatabase.Exceptions.OdbRuntimeException);
```

Inheritance Hierarchy

System.
 System.
  NDatabase.Exceptions.
   NDatabase.Exceptions.

See Also

CorruptedDatabaseException Members
NDatabase.Exceptions Namespace

9.5.3.1 CorruptedDatabaseException Members

The CorruptedDatabaseException type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)
	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also

© 2014 Jacek Spolnik

CorruptedDatabaseException Class
NDatabase.Exceptions Namespace

9.5.3.2 CorruptedDatabaseException Methods

The CorruptedDatabaseException type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

See Also
CorruptedDatabaseException Class
NDatabase.Exceptions Namespace

9.5.3.3 CorruptedDatabaseException Properties

The CorruptedDatabaseException type exposes the following members.

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)
	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also
CorruptedDatabaseException Class
NDatabase.Exceptions Namespace

9.5.4 DuplicatedKeyException Class

Exception raised when error in BTrees will appear (Duplicated key)

Namespace: NDatabase.Exceptions
Assembly: NDatabase3 (in NDatabase3.dll)

© 2014 Jacek Spolnik

Syntax

Visual Basic

```
Public NotInheritable Class DuplicatedKeyException _
 Inherits BTreeException
```

C#

```
public sealed class DuplicatedKeyException : BTreeException
```

Visual C++

```
public refclass DuplicatedKeyException sealed : public BTreeException
```

JavaScript

```
NDatabase.Exceptions.DuplicatedKeyException = function();
Type.createClass(
 'NDatabase.Exceptions.DuplicatedKeyException',
 NDatabase.Exceptions.BTreeException);
```

Inheritance Hierarchy

System.
 System.
  NDatabase.Exceptions.
   NDatabase.Exceptions.
    NDatabase.Exceptions.

See Also

DuplicatedKeyException Members
NDatabase.Exceptions Namespace

9.5.4.1 DuplicatedKeyException Members

The DuplicatedKeyException type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)

© 2014 Jacek Spolnik

	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also
DuplicatedKeyException Class
NDatabase.Exceptions Namespace

9.5.4.2 DuplicatedKeyException Methods

The DuplicatedKeyException type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

See Also
DuplicatedKeyException Class
NDatabase.Exceptions Namespace

9.5.4.3 DuplicatedKeyException Properties

The DuplicatedKeyException type exposes the following members.

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)
	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

© 2014 Jacek Spolnik

See Also
DuplicatedKeyException Class
NDatabase.Exceptions Namespace

9.5.5 LinqQueryException Class

NDatabase exception raised during processing linq query

Namespace: NDatabase.Exceptions
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public NotInheritable Class LinqQueryException _
 Inherits OdbRuntimeException
```

C#

```
public sealed class LinqQueryException : OdbRuntimeException
```

Visual C++

```
public ref class LinqQueryException sealed : public OdbRuntimeException
```

JavaScript

```
NDatabase.Exceptions.LinqQueryException = function();
Type.createClass(
 'NDatabase.Exceptions.LinqQueryException',
 NDatabase.Exceptions.OdbRuntimeException);
```

Inheritance Hierarchy

System.
 System.
  NDatabase.Exceptions.
   NDatabase.Exceptions.

See Also
LinqQueryException Members
NDatabase.Exceptions Namespace

9.5.5.1 LinqQueryException Members

The LinqQueryException type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

© 2014 Jacek Spolnik

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)
	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also
LinqQueryException Class
NDatabase.Exceptions Namespace

9.5.5.2 LinqQueryException Methods

The LinqQueryException type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

See Also
LinqQueryException Class
NDatabase.Exceptions Namespace

9.5.5.3 LinqQueryException Properties

The LinqQueryException type exposes the following members.

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)

© 2014 Jacek Spolnik

	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also
LinqQueryException Class
NDatabase.Exceptions Namespace

9.5.6 OdbRuntimeException Class

Generic ODB Runtime exception : Used to report all problems.

Namespace: NDatabase.Exceptions
Assembly: NDatabase3 (in NDatabase3.dll)

Syntax

Visual Basic

```
Public Class OdbRuntimeException _
 Inherits Exception
```

C#

```
public class OdbRuntimeException : Exception
```

Visual C++

```
public ref class OdbRuntimeException : public Exception
```

JavaScript

```
NDatabase.Exceptions.OdbRuntimeException = function();
Type.createClass(
 'NDatabase.Exceptions.OdbRuntimeException',
 Exception);
```

Inheritance Hierarchy

System.
 System.
  NDatabase.Exceptions.
   NDatabase.Exceptions.
   NDatabase.Exceptions.
   NDatabase.Exceptions.
   NDatabase.Exceptions.

See Also
OdbRuntimeException Members
NDatabase.Exceptions Namespace

9.5.6.1 OdbRuntimeException Members

The OdbRuntimeException type exposes the following members.

Methods

	Name	Description

© 2014 Jacek Spolnik

	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)
	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also
OdbRuntimeException Class
NDatabase.Exceptions Namespace

9.5.6.2 OdbRuntimeException Methods

The OdbRuntimeException type exposes the following members.

Methods

	Name	Description
	Equals	(Inherited from Object.)
	Finalize	(Inherited from Object.)
	GetBaseException	(Inherited from Exception.)
	GetHashCode	(Inherited from Object.)
	GetObjectData	(Inherited from Exception.)
	GetType	(Inherited from Exception.)
	MemberwiseClone	(Inherited from Object.)
	ToString	(Inherited from Exception.)

See Also
OdbRuntimeException Class
NDatabase.Exceptions Namespace

© 2014 Jacek Spolnik

9.5.6.3 OdbRuntimeException Properties

The OdbRuntimeException type exposes the following members.

Properties

	Name	Description
	Data	(Inherited from Exception.)
	HelpLink	(Inherited from Exception.)
	HResult	(Inherited from Exception.)
	InnerException	(Inherited from Exception.)
	Message	(Inherited from Exception.)
	Source	(Inherited from Exception.)
	StackTrace	(Inherited from Exception.)
	TargetSite	(Inherited from Exception.)

See Also

OdbRuntimeException Class
NDatabase.Exceptions Namespace

© 2014 Jacek Spolnik