SphinxConnector.NET v5.0 (21 Feb 2021) + CRACK

2025-01-06 23:58:40 label 我要反馈 下载页面

SphinxConnector.NET is a .NET client library to get the features and functions of the Sphinx full-text search engine out of any .NET program.

SphinxConnector.NET is an all-purpose .NET client library to get the features and functions of the Sphinx full-text search engine out of any .NET program.

It supplies an eloquent query API which makes utilizing Sphinx with. NET simpler and more comfortable than ever before. Its eloquent interface was designed for great efficacy and simplicity of usage. Operating directly in your record versions permits you to produce your inquiries in a strongly typed, LINQ-like style.

SphinxConnector.NET is totally asynchronous. NET client API for its Sphinx full-text search engine along with its own fork Manticore Search. It supplies an eloquent query API with object mapping capacities that allow for easy Sphinx integration to your own applications. Following the setup, SphinxConnector.NET works in a trial mode, which has a few limitations in performance.

SphinxConnector.NET Great Features:

The fluent API provides you with a LINQ-like query API to design your full-text queries. It is tailored to the needs of .NET developers working with the Sphinx full-text search engine. Naturally, the fluent API supports tasks like ordering and grouping results, defining result-set sizes, and projecting your results into, e.g., anonymous types. In addition to querying, it also supports saving and deleting documents from real-time indexes for which it generates optimized statements depending on the context.

Under the hood, SphinxConnector.NET translates your input to SphinxQL, executes your query using its infrastructure for SphinxQL, and maps the results to your document model. UsinUsinghinxQL API as the underlying query mechanism, users benefit from features like connection pooling without manually writing.

To start querying, you have to create a class representing your documents, and you are good to go.

Querying Sphinx indexes is done via the IFulltextSession interface. It provides a Query method that takes your document model as a generic type argument and returns an instance of IFulltextQuery.

The IFulltextQuery interface provides you with all the methods you need to order easily, group, or filter your documents. SphinxConnector.NET automatically translates calls to supported methods and properties of .NET types to their equivalent SphinxQL functions. Supported are, for example, methods of the System. Math class and properties of System.DateTime. Additionally, SphinxConnector.NET defines extension methods like In and Interval for use in your queries.

All fluent API methods have both synchronous and asynchronous implementations.

With the release of the version, Sphinx introduced a new attribute type that allows users to store JSON objects in an index. SphinxConnector.NET supports these attributes since version (via Json.NET at that time, since System.Text.Json has replaced version Json.NET).

Future queries provide an easy way to optimize your performance in scenarios where multiple queries need to be executed. Future queries are only executed when their results are accessed. This allows SphinxConnector.NET to send multiple queries in a single network round-trip to the Sphinx server.

Saving & Deleting Documents

To save a document to a real-time index, you only need to pass an instance of your document model to the Save method of

标签

Other .Net Components

the IFulltextSession interface.

To delete one or more documents from a real-time index, you can pass their IDs to the Delete method or an instance of a document you would like to delete.

As you can see from the examples, you call FlushChanges to let SphinxConnector.NET know that the changes you made should be made persistent.

Automatic Optimization and Batching

When you call FlushChanges, all pending saves and deletes are gathered and executed most efficiently. All deletes are executed in a single DELETE statement for the example above, thus avoiding unnecessary network round-trips.

When SphinxConnector.NET detects that it needs to save more than one document, it inserts them in batches by generating a single REPLACE statement for each batch. This leads to a speed-up by several orders of magnitude compared to inserting documents one by one. The batch size is, of course, configurable so that you can fine-tune it to your workload.

Convention over Configuration

SphinxConnector.NET uses a convention-based approach for many of its configuration settings. For example, when mapping objects to indexes or properties to attributes, you do not have to provide any manual mappings or clutter your code with custom attributes. The mapping is done via conventions that are defined in the FulltextStore class. You can, of course, change the default conventions if necessary.

Notice that the conventions are defined via delegates. Often you have to implement interfaces or inherit from base classes to change some behavior in a component that you are using. Most of the time, we think this poses an unnecessary overhead because the customizations are just a few lines of code. In these cases providing a simple delegate saves time, keeps the code clean and easy to maintain. And in more complex scenarios, you have the option to implement abstractions the way you see fit without being limited or constrained in your design by some 3rd party interfaces.

Object Mapping for SphinxQL

If needed, you can also execute SphinxQL queries directly and have SphinxConnector.NET map the results to your document objects.

Designed for Testability

All functionality of the fluent API is exposed via interfaces. By applying techniques such as Inversion of Control and Dependency Injection, you can easily test your code by substituting these interfaces with mocks if necessary.

Using the fluent API with your favorite DI container is also straightforward:

Classic API's

SphinxConnector.NET provides high-performance implementations of the native API and standard ADO.NET 2.0 classes for executing SphinxQL statements. Please see below for a list of features exposed by each API:

SphinxQL

- Based on standard ADO.NET 2.0 classes:
  - SphinxQLConnection
  - SphinxQLConnectionStringBuilder
  - SphinxQLCommand
  - SphinxQLDataAdapter
  - SphinxQLDataReader
  - SphinxQLParameter
  - SphinxQLTransaction
- Fully asynchronous
- Tailored to Sphinx specific data types and features, e.g.
  - Automatic conversion from .NET types to Sphinx types and vice versa
  - Methods for handling MVA values
- Insert and Update Records in Real-Time Indexes
- Select from any Index Type
- Select via DataReader or DataAdapter
- Connection Pooling for Maximum Performance
- Support for Transactions
- Support for TransactionScope (Local and Distributed Transactions)
- Support for Command Parameters
- Automatically detects and uses features based on Sphinx version, e.g., server-side multi-query support with Sphinx 2.0.1 and above

Native API*

- Support for Sphinx 2.0.1 and up
- Support for string attributes introduced with Sphinx 1.10.1
- Support for features introduced in 0.9.9 like:
  - Persistent connections: open one connection for several operations to minimize network overhead.
  - Override attributes: temporarily change the value of an attribute without modifying the actual value for advanced search scenarios.
  - Select clause: write SQL-like statements to operate on attributes
- Configurable Encoding

- Search related features:
  - Set value and range filters
  - Schedule several queries for batch execution
  - Specify how Sphinx should match and rank documents
  - and more
- Access to additional functions exposed by the Sphinx search engine:
  - Build excerpts
  - Build keywords
  - Update attributes
  - Query Sphinx for status variables
- SphinxConnector.NET uses custom types as arguments for methods operating on Sphinx attributes. This gives you strongly typed access to these methods and also enabled the compiler to check whether the operation is valid for a given type.

*Please note that the native API is only intended to support legacy projects and be removed in a future version. New projects should use the fluent API or SphinxQL.

Logging

SphinxConnector.NET supports logging different types of messages to help developers identify problems or optimize their setup. Since the release of V5, SphinxConnector.NET provides its own simple logging abstraction, replacing the previously used Common. Logging library. A simple Console logger is included in the main library, additional NLog, log4net, Serilog, and Microsoft.Extensions.Logging is available via NuGet.

资源列表

download SphinxConnector.NET v5.0 (21 Feb 2021)