MySqlBackup.NET - MySQL Backup Solution for C#, VB.NET, ASP.NET

adriancs

26 Jun 2019 Public Domain

A tool to export and import MySQL database in .NET

Download source files - 5.5 MB

Available at:

- https://github.com/MySqlBackupNET/MySqlBackup.Net
- ASP.NET Online Demo: http://mysqlbackup.somee.com
- ohloh.net - https://www.ohloh.net/p/MySqlBackupNET
- Install via NuGet: **PM> Install-Package MySqlBackup.NET**
  https://www.nuget.org/packages/MySqlBackup.NET/[^]
  What is NuGet? | *How do you install a NuGet package in Visual Studio 2012 ?*

Contents

1. Introduction

This article introduces a tool (DLL) that can backup/restore MySQL database in C# or VB.NET and some sample codes on how to use it. It is an alternative to `MySqlDump`.

Another benefits of making this tool is, we don't have to rely on two small programs - *MySqlDump.exe* and *MySql.exe* to perform the backup and restore task. We will have better control on the output result.

The most common way to backup a MySQL Database is by using *MySqlDump.exe* and **MySQL Workbench**.

MySQL Workbench is good for developers, but, when it comes to client or end-user, the recommended way is to get every parameter preset and all they need to know is press the big button "**Backup**" and everything is done. Using *MySQL Workbench* as a backup tool is not a suitable solution for client or end-user.

On the other hand, *MySqlDump.exe* cannot be used for Web applications. As most web hosting providers forbid that, *MySqlBackup.NET* will be helpful in building a web-based (ASP.NET/Web-Services) backup tool.

2. Features & Dependencies

Features

- Backup and Restore of MySQL Database
- Can be used in any .NET Language
- Export/Import to/from `MemoryStream`
- Conditional Rows Export (Filter Tables or Rows)
- Progress Report is Available for Both Export and Import Task
- Able to export rows into different modes (Insert, Insert Ignore, Replace, On Duplicate Key Update, Update)
- Can be used directly in ASP.NET or web services

Prerequisite / Dependencies

MySqlBackup.NET is built on top of MySQL dot net Connector/Net (MySql.Data.DLL)

- A reference of this DLL must be added into your project in order for MySqlBackup.NET to work
- *MySql.Data.DLL* is developed by Oracle Corporation, licensed under GPL License (http://www.gnu.org/licenses/old-licenses/gpl-2.0.html)

3. Background

This article assumes that you are already familiar with MySQL dot net connector (*MySql.Data.dll*) with minimum knowledge that you are able to perform the four basic operations, `SELECT`, `INSERT`, `UPDATE`, and `DELETE`. In case you are not, you can read the walk-through and explanation on Connecting C# to MySQL at: [http://www.codeproject.com/Articles/43438/Connect-C-to-MySQL].

4. Basic Usage

Add this `using` statement before coding with MySqlBackup.NET:

```
using MySql.Data.MySqlClient;
```

Simple Export Example

```
string constring = "server=localhost;user=root;pwd=qwerty;database=test;";
string file = "C:\\backup.sql";
using (MySqlConnection conn = new MySqlConnection(constring))
{
    using (MySqlCommand cmd = new MySqlCommand())
    {
        using (MySqlBackup mb = new MySqlBackup(cmd))
        {
            cmd.Connection = conn;
            conn.Open();
            mb.ExportToFile(file);
            conn.Close();
        }
    }
}
```

Simple Import Example

```
string constring = "server=localhost;user=root;pwd=qwerty;database=test;";
string file = "C:\\backup.sql";
using (MySqlConnection conn = new MySqlConnection(constring))
{
    using (MySqlCommand cmd = new MySqlCommand())
    {
        using (MySqlBackup mb = new MySqlBackup(cmd))
        {
            cmd.Connection = conn;
            conn.Open();
            mb.ImportFromFile(file);
            conn.Close();
        }
    }
}
```

The above examples will export and import a MySQL database with default options. There are some options that can modify the export and import behavior. These options are defined in:

- `MySqlBackup.ExportInfo`
- `MySqlBackup.ImportInfo`

Example of customize `export` behavior:

- Create new database
- Only export table's structures
- Don't export rows of data

Sample Codes:

```
string constring = "server=localhost;user=root;pwd=1234;database=test1;";
string file = "Y:\\backup.sql";
using (MySqlConnection conn = new MySqlConnection(constring))
{
    using (MySqlCommand cmd = new MySqlCommand())
    {
        using (MySqlBackup mb = new MySqlBackup(cmd))
        {
            cmd.Connection = conn;
            conn.Open();
            mb.ExportInfo.AddCreateDatabase = true;
            mb.ExportInfo.ExportTableStructure = true;
            mb.ExportInfo.ExportRows = false;
            mb.ExportToFile(file);
        }
    }
}
```

Full List of ExportInfo Options

```
ExportInfo.GetDocumentHeaders(cmd)
```

- return: `List<string>`
- default value: Demonstrated in test app

- Gets the list of document headers

```
ExportInfo.SetDocumentHeaders(List<string>)
```

- Sets the document headers

```
ExportInfo.GetDocumentFooters()
```

- return: `List<string>`
- default value: demonstrated in test app
- Gets the document footers

```
ExportInfo.SetDocumentFooters(List<string>)
```

- Sets the document headers

```
ExportInfo.ExcludeTables - List<string>
```

- default value: empty list
- Gets or Sets the tables (black list) that will be excluded for export. The rows of these tables will not be exported too.

```
ExportInfo.TablesToBeExportedList - List<string>
```

- default value: empty list
- Gets or Sets the list of tables that will be exported. If none, all tables will be exported.

```
ExportInfo.TablesToBeExportedDic - Dictionary<string, string>
```

- default value: empty dictionary
- Gets or Sets the tables that will be exported with custom `SELECT` defined
- If none or empty, all tables and rows will be exported
- Key = Table's Name. Value = Custom `SELECT` Statement
- Example 1: `SELECT * FROM product WHERE category = 1;`
- Example 2: `SELECT name,description FROM product;`

```
ExportInfo.RecordDumpTime - bool
```

- default value: `true`
- Gets or Sets a value indicates whether the Dump Time should be recorded in dump file

```
ExportInfo.AddCreateDatabase - bool
```

- default value: `false`
- Gets or Sets a value indicates whether the SQL statement of "`CREATE DATABASE`" should be added into dump file.

```
ExportInfo.AddDropDatabase - bool
```

- default value: `false`
- Gets or Sets a value indicates whether the SQL statement of "`DROP DATABASE`" should be added into dump file

```
ExportInfo.ExportTableStructure - bool
```

- default value: `true`
- Gets or Sets a value indicates whether the Table Structure (`CREATE TABLE`) should be exported.

```
ExportInfo.AddDropTable - bool
```

- default value: `true`
- Gets or Sets a value indicates whether the SQL statement of "`DROP TABLE`" should be added into the dump file

```
ExportInfo.ResetAutoIncrement - bool
```

- default value: `false`
- Gets or Sets a value indicates whether the value of auto-increment of each table should be reset to `1`

```
ExportInfo.ExportRows - bool
```

- default value: `true`
- Gets or Sets a value indicates whether the Rows should be exported.

```
ExportInfo.MaxSqlLength - int
```

- default value: 5 * 1024 * 1024 (5mb)
- Gets or Sets the maximum length for combining multiple `INSERT`s into single SQL
- Default value is 5MB.
- Only applies if `RowsExportMode = "INSERT" or "INSERTIGNORE" or "REPLACE"`
- This value will be ignored if `RowsExportMode = ONDUPLICATEKEYUPDATE or UPDATE`

```
ExportInfo.ExportProcedures - bool
```

- default value: `true`
- Gets or Sets a value indicates whether the Stored Procedures should be exported

```
ExportInfo.ExportFunctions - bool
```

- default value: `true`
- Gets or Sets a value indicates whether the Stored Functions should be exported

```
ExportInfo.ExportTriggers - bool
```

- default value: `true`
- Gets or Sets a value indicates whether the Stored Triggers should be exported

```
ExportInfo.ExportViews - bool
```

- default value: `true`
- Gets or Sets a value indicates whether the Stored Views should be exported

```
ExportInfo.ExportEvents - bool
```

- default value: `true`
- Gets or Sets a value indicates whether the Stored Events should be exported

```
ExportInfo.IntervalForProgressReport - int
```

- default value: `100`
- Gets or Sets a value indicates the interval of time (in miliseconds) to raise the event of `ExportProgressChanged`

```
ExportInfo.ScriptsDelimiter - string
```

- default value: `|`
- Gets or Sets the delimiter used for exporting Procedures, Functions, Events and Triggers

```
ExportInfo.ExportRoutinesWithoutDefiner - bool
```

- default value: `true`
- Gets or Sets a value indicates whether the exported Scripts (Procedure, Functions, Events, Triggers, Events) should exclude `DEFINER`

```
ExportInfo.RowsExportMode - enum RowsDataExportMode
```

- default value: `Insert`
- Gets or Sets an `enum` value indicates how the rows of each table should be exported
- `INSERT` = The default option. Recommended if exporting to a new database. If the primary key existed, the process will halt.
- `INSERT IGNORE` = If the primary key existed, skip it
- `REPLACE` = If the primary key existed, delete the row and insert new data
- `OnDuplicateKeyUpdate` = If the primary key existed, update the row. If all fields are primary keys, it will change to `INSERT IGNORE`.
- `UPDATE` = If the primary key does not exist, skip it and if all the fields are primary key, no rows will be exported.

```
ExportInfo.WrapWithinTransaction - bool
```

- default value: `false`
- Gets or Sets a value indicates whether the rows dump should be wrapped with transaction.
- Recommended to set this value to `FALSE` if using `RowsExportMode = "INSERT" or "INSERTIGNORE" or "REPLACE", else TRUE.`

```
ExportInfo.TextEncoding - System.Text.Encoding
```

- default value: `UTF8Encoding(false)`
- Gets or Sets a value indicates the encoding to be used for exporting the dump.

```
ExportInfo.BlobExportMode - enum BlobDataExportMode
```

- default value: `BlobDataExportMode.HexString`
- Gets or Sets an `enum` value indicates how the BLOB should be exported.
- `BinaryChar` = `char` format
- **Note**: Export BLOB into Binary Char is not intended for real deploy usage at the moment. Exporting into `BinaryChar` will raise an exception which attempts to alarm the developers that this function is meant for development and debugging purposes. Read more: https://github.com/MySqlBackupNET/MySqlBackup.Net/issues/47

```
ExportInfo.BlobExportModeForBinaryStringAllow - bool
```

- default value: `false`
- If you wish to help to debug, fix or develop the function of exporting BLOB into binary char format (`BlobExportMode=BinaryChar`), set this value to `true`

```
ExportInfo.GetTotalRowsMode - enum GetTotalRowsMethod
```

- default value: `InformationSchema`
- Gets or Sets a value indicates the method of how the total rows value is being obtained
- This function is useful if you are developing a progress bar
- `InformationSchema` = Fast, but approximate value
- `SelectCount` = Slow but accurate
- `Skip` = Skip obtaining total rows. Use this option if you are not doing any progress report.

Full List of ImportInfo Options

```
ImportInfo.IntervalForProgressReport - int
```

- default value: `100`
- Gets or Sets a value indicates the interval of time (in milliseconds) to raise the event of `ExportProgressChanged`

```
ImportInfo.IgnoreSqlError - bool
```

- default value: `false`
- Gets or Sets a value indicates whether SQL errors occurs in import process should be ignored

```
ImportInfo.ErrorLogFile - string
```

- default value: `string.empty`
- Gets or Sets the file path used to log error messages

5. Example of Using in ASP.NET

Sample code for Export. The below codes will export the content into `MemoryStream`, then transmit it directly for download.

```
using System.IO;

string connstr = "server=localhost;user=root;pwd=1234;database=test;";
MemoryStream ms = new MemoryStream();
using (MySqlConnection conn = new MySqlConnection(connstr))
{
    MySqlCommand cmd = new MySqlCommand();
    MySqlBackup mb = new MySqlBackup(cmd);
    cmd.Connection = conn;
    conn.Open();
    mb.ExportToMemoryStream(ms);
}
Response.ContentType = "text/plain";
Response.AppendHeader("Content-Disposition", "attachment; filename=backup.sql");
Response.BinaryWrite(ms.ToArray());
Response.End();
```

Sample code for Upload and Import:

```
string connstr = "server=localhost;user=root;pwd=1234;database=test;";
byte[] ba = FileUpload1.FileBytes;
MemoryStream ms = new MemoryStream(ba);
using (MySqlConnection conn = new MySqlConnection(connstr))
{
    MySqlCommand cmd = new MySqlCommand();
    MySqlBackup mb = new MySqlBackup(cmd);
    cmd.Connection = conn;
    conn.Open();
    mb.ExportToMemoryStream(ms);
}
Header.Controls.Add(new LiteralControl
     ("<script type=\"text/javascript\">alert('ok');</script>"));
```

6. More Guides And Examples

More guides and examples are available at the project site's documentation:
https://github.com/adriancs2/MySqlBackup.Net/wiki[^]

Below are some of the guides.

- Example of Using in MemoryStream, Zip and ASP.NET
- Using Progress Report With Export/Backup
- Using Progress Report With Import/Restore
- Conditional Rows Export for Each Table
- FAQ - Commonly Seen Errors

7. History

- June 6, 2019, Release of v2.3
- View full change log

License

About the Author

adriancs
Software Developer
Malaysia

Programming is an art.

Comments and Discussions

325 messages have been posted for this article Visit **https://www.codeproject.com/Articles/256466/MySqlBackup-NET** to post and view comments on this article, or click **here** to get a print view with messages.

Permalink
Advertise
Privacy
Cookies
Terms of Use