免費電子書

學習

Dapper.NET

Free unaffiliated eBook created from
Stack Overflow contributors.

#dapper

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: dapper-net

It is an unofficial and free Dapper.NET ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Dapper.NET.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: Dapper.NET

Dapper

Dapper.NetORM`IDbConnection`。

- github https //github.com/StackExchange/dapper-dot-net
- NuGet https //www.nuget.org/packages/Dapper

-
-

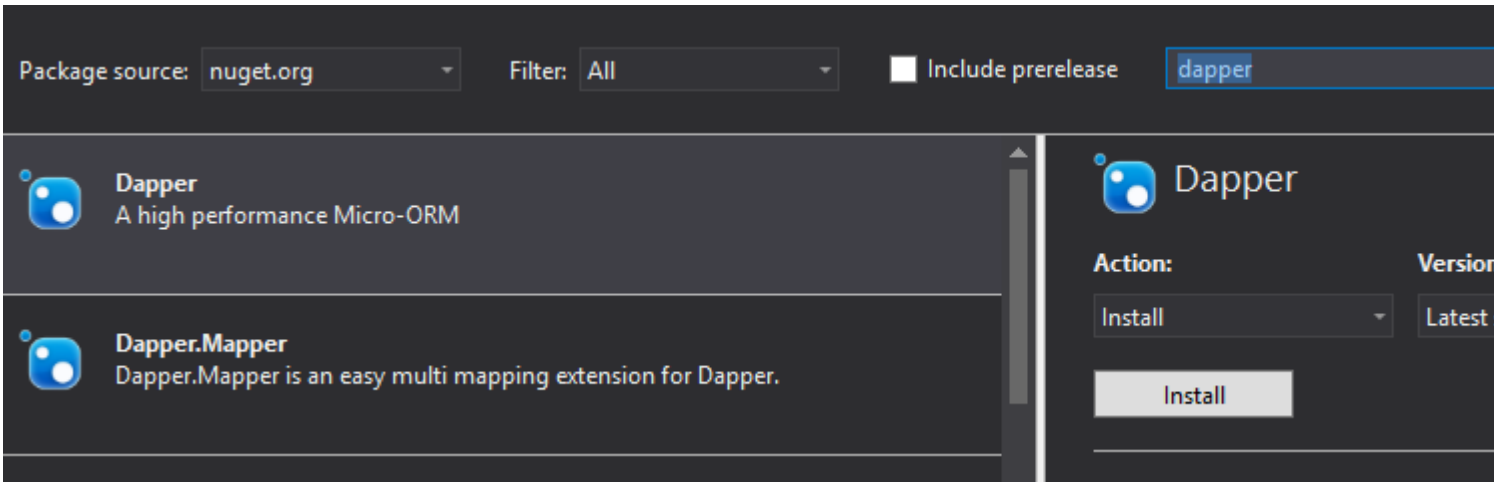
1.50.0	core-clr / asp.net 5.0RTM	2016629
1.42.0		201556
1.40.0		201543
1.30.0		2014814
1.20.0		201458
1.10.0		2012-06-27
1.0.0		2011-04-14

Examples

NugetDapper

Visual Studio GUI

> NuGet>...Visual Studio 2015



Nuget Power Shell

```
Install-Package Dapper
```

```
Install-Package Dapper -Version 1.42.0
```

CDapper

```
using System.Data;
using System.Linq;
using Dapper;

class Program
{
    static void Main()
    {
        using (IDbConnection db = new
SqlConnection("Server=myServer;Trusted_Connection=true"))
        {
            db.Open();
            var result = db.Query<string>("SELECT 'Hello World'").Single();
            Console.WriteLine(result);
        }
    }
}
```

“ Using

LINQPadDapper

LINQPadNuGet 。 LINQPadDapper**F4**“”“ **NuGet”** 。 **dapper dot netAdd To Query** 。 Dapper LINQPad。

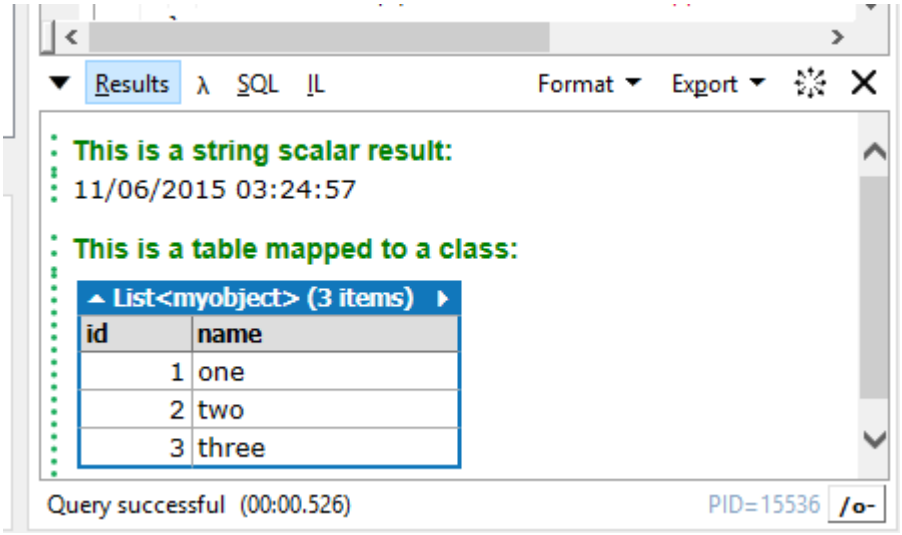
Dapper**C** C.Dump

```
void Main()
{
    using (IDbConnection db = new SqlConnection("Server=myServer;Trusted_Connection=true")){
```

```
        db.Open();
        var scalar = db.Query<string>("SELECT GETDATE()").SingleOrDefault();
        scalar.Dump("This is a string scalar result:");

        var results = db.Query<myobject>(@"
        SELECT * FROM (
        VALUES (1,'one'),
            (2,'two'),
            (3,'three')
        ) AS mytable(id,name)");
        results.Dump("This is a table mapped to a class:");
    }
}

// Define other methods and classes here
class myobject {
    public int id { get; set; }
    public string name { get; set; }
}
```



Dapper.NET https://riptutorial.com/zh-TW/dapper/topic/2/dapper-net

2: Multimapping

- `public static IEnumerable<TReturn> Query<TFirst, TSecond, TReturn>(this IDbConnection cnn, string sql, Func<TFirst, TSecond, TReturn> map, object param = null, IDbTransaction transaction = null, bool buffered = true, string splitOn = "Id", int? commandTimeout = null, CommandType? commandType = null)`
- `public static IEnumerable<TReturn> Query<TFirst, TSecond, TThird, TFourth, TFifth, TSixth, TSeventh, TReturn>(this IDbConnection cnn, string sql, Func<TFirst, TSecond, TThird, TFourth, TFifth, TSixth, TSeventh, TReturn> map, object param = null, IDbTransaction transaction = null, bool buffered = true, string splitOn = "Id", int? commandTimeout = null, CommandType? commandType = null)`
- `public static IEnumerable<TReturn> Query<TReturn>(this IDbConnection cnn, string sql, Type[] types, Func<object[], TReturn> map, object param = null, IDbTransaction transaction = null, bool buffered = true, string splitOn = "Id", int? commandTimeout = null, CommandType? commandType = null)`

	。
SQL	。
	。
	`Func<>` 。
PARAM	。
	。
	。 true。 bufferedtrue`List<T> IEnumerable<T>`。 bufferedfalsesql。 。 buffered false 。 sql。
splitOn	id。 。
CommandTimeout	。

Examples

Person。

	1942	
	1967	
	1941	

```
public class Person
```

```
{
    public string Name { get; set; }
    public int Born { get; set; }
    public Country Residience { get; set; }
}

public class Country
{
    public string Residence { get; set; }
}
```

Func<>Query<>personResidenceFunc<>。 Func<>7。

```
var sql = @"SELECT 'Daniel Dennett' AS Name, 1942 AS Born, 'United States of America' AS
Residence
UNION ALL SELECT 'Sam Harris' AS Name, 1967 AS Born, 'United States of America' AS Residence
UNION ALL SELECT 'Richard Dawkins' AS Name, 1941 AS Born, 'United Kingdom' AS Residence";

var result = connection.Query<Person, Country, Person>(sql, (person, country) => {
        if(country == null)
        {
            country = new Country { Residence = "" };
        }
        person.Residience = country;
        return person;
    },
    splitOn: "Residence");
```

splitOn: "Residence"Country。 Dapper*Id*splitOnSystem.ArgumentException。 splitOn。

。 。 。

。

ID			CountryId		BOOKID	BOOKNAME
1		1942	1		1	
1		1942	1		2	
2		1967	1		3	
2		1967	1		4	
3		1941	2			
3		1941	2		6	

```
public class Person
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Born { get; set; }
    public Country Residience { get; set; }
```

```
    public ICollection<Book> Books { get; set; }
}

public class Country
{
    public int CountryId { get; set; }
    public string CountryName { get; set; }
}

public class Book
{
    public int BookId { get; set; }
    public string BookName { get; set; }
}
```

`remainingHorsemen` lambda

```
            var sql = @"SELECT 1 AS Id, 'Daniel Dennett' AS Name, 1942 AS Born, 1 AS
CountryId, 'United States of America' AS CountryName, 1 AS BookId, 'Brainstorms' AS BookName
UNION ALL SELECT 1 AS Id, 'Daniel Dennett' AS Name, 1942 AS Born, 1 AS CountryId, 'United
States of America' AS CountryName, 2 AS BookId, 'Elbow Room' AS BookName
UNION ALL SELECT 2 AS Id, 'Sam Harris' AS Name, 1967 AS Born, 1 AS CountryId,  'United States
of America' AS CountryName, 3 AS BookId, 'The Moral Landscape' AS BookName
UNION ALL SELECT 2 AS Id, 'Sam Harris' AS Name, 1967 AS Born, 1 AS CountryId,  'United States
of America' AS CountryName, 4 AS BookId, 'Waking Up: A Guide to Spirituality Without Religion'
AS BookName
UNION ALL SELECT 3 AS Id, 'Richard Dawkins' AS Name, 1941 AS Born, 2 AS CountryId,  'United
Kingdom' AS CountryName, 5 AS BookId, 'The Magic of Reality: How We Know What`s Really True'
AS BookName
UNION ALL SELECT 3 AS Id, 'Richard Dawkins' AS Name, 1941 AS Born, 2 AS CountryId,  'United
Kingdom' AS CountryName, 6 AS BookId, 'An Appetite for Wonder: The Making of a Scientist' AS
BookName";

var remainingHorsemen = new Dictionary<int, Person>();
connection.Query<Person, Country, Book, Person>(sql, (person, country, book) => {
    //person
    Person personEntity;
    //trip
    if (!remainingHorsemen.TryGetValue(person.Id, out personEntity))
    {
        remainingHorsemen.Add(person.Id, personEntity = person);
    }

    //country
    if(personEntity.Residience == null)
    {
        if (country == null)
        {
            country = new Country { CountryName = "" };
        }
        personEntity.Residience = country;
    }

    //books
    if(personEntity.Books == null)
    {
        personEntity.Books = new List<Book>();
    }

    if (book != null)
```

```
    {
        if (!personEntity.Books.Any(x => x.BookId == book.BookId))
        {
            personEntity.Books.Add(book);
        }
    }

    return personEntity;
},
splitOn: "CountryId,BookId");
```

splitOn◦

7

Func <>◦

Query<> ◦ ◦

```
            var sql = @"SELECT 1 AS Id, 'Daniel Dennett' AS Name, 1942 AS Born, 1 AS
CountryId, 'United States of America' AS CountryName, 1 AS BookId, 'Brainstorms' AS BookName
UNION ALL SELECT 1 AS Id, 'Daniel Dennett' AS Name, 1942 AS Born, 1 AS CountryId, 'United
States of America' AS CountryName, 2 AS BookId, 'Elbow Room' AS BookName
UNION ALL SELECT 2 AS Id, 'Sam Harris' AS Name, 1967 AS Born, 1 AS CountryId,  'United States
of America' AS CountryName, 3 AS BookId, 'The Moral Landscape' AS BookName
UNION ALL SELECT 2 AS Id, 'Sam Harris' AS Name, 1967 AS Born, 1 AS CountryId,  'United States
of America' AS CountryName, 4 AS BookId, 'Waking Up: A Guide to Spirituality Without Religion'
AS BookName
UNION ALL SELECT 3 AS Id, 'Richard Dawkins' AS Name, 1941 AS Born, 2 AS CountryId,  'United
Kingdom' AS CountryName, 5 AS BookId, 'The Magic of Reality: How We Know What`s Really True'
AS BookName
UNION ALL SELECT 3 AS Id, 'Richard Dawkins' AS Name, 1941 AS Born, 2 AS CountryId,  'United
Kingdom' AS CountryName, 6 AS BookId, 'An Appetite for Wonder: The Making of a Scientist' AS
BookName";

var remainingHorsemen = new Dictionary<int, Person>();
connection.Query<Person>(sql,
    new[]
    {
        typeof(Person),
        typeof(Country),
        typeof(Book)
    }
    , obj => {

        Person person = obj[0] as Person;
        Country country = obj[1] as Country;
        Book book = obj[2] as Book;

        //person
        Person personEntity;
        //trip
        if (!remainingHorsemen.TryGetValue(person.Id, out personEntity))
        {
            remainingHorsemen.Add(person.Id, personEntity = person);
        }

        //country
```

```
        if(personEntity.Residience == null)
        {
            if (country == null)
            {
                country = new Country { CountryName = "" };
            }
            personEntity.Residience = country;
        }

        //books
        if(personEntity.Books == null)
        {
            personEntity.Books = new List<Book>();
        }

        if (book != null)
        {
            if (!personEntity.Books.Any(x => x.BookId == book.BookId))
            {
                personEntity.Books.Add(book);
            }
        }

        return personEntity;
},
splitOn: "CountryId,BookId");
```

◦ System.Data.Linq.Mapping.ColumnAttribute◦

◦

```
public class Person
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Born { get; set; }
    public Country Residience { get; set; }
    public ICollection<Book> Books { get; set; }
}

public class Country
{
    [System.Data.Linq.Mapping.Column(Name = "CountryId")]
    public int Id { get; set; }

    [System.Data.Linq.Mapping.Column(Name = "CountryName")]
    public string Name { get; set; }
}

public class Book
{
    public int Id { get; set; }

    public string Name { get; set; }
}
```

BookColumnAttributeif

```
Dapper.SqlMapper.SetTypeMap(
    typeof(Country),
    new CustomPropertyTypeMap(
        typeof(Country),
        (type, columnName) =>
            type.GetProperties().FirstOrDefault(prop =>
                prop.GetCustomAttributes(false)
                    .OfType<System.Data.Linq.Mapping.ColumnAttribute>()
                    .Any(attr => attr.Name == columnName)))
);


var bookMap = new CustomPropertyTypeMap(
    typeof(Book),
    (type, columnName) =>
    {
        if(columnName == "BookId")
        {
            return type.GetProperty("Id");
        }

        if (columnName == "BookName")
        {
            return type.GetProperty("Name");
        }

        throw new InvalidOperationException($"No matching mapping for {columnName}");
    }
);
Dapper.SqlMapper.SetTypeMap(typeof(Book), bookMap);
```

Query<>Query<> ◦

◦

Multimapping https://riptutorial.com/zh-TW/dapper/topic/351/multimapping

3:

- conn.Executesqltransactiontran; //
- conn.Executesqlparameterstran;
- conn.Querysqltransactiontran;
- conn.Querysqlparameterstran;
- await conn.ExecuteAsyncsqltransactiontran; //
- await conn.ExecuteAsyncsqlparameterstran;
- await conn.QueryAsyncsqltransactiontran;
- await conn.QueryAsyncsqlparameterstran;

Examples

SqlConnectionIDbConnection。

IDbConnectionIDbTransaction。

```
public void UpdateWidgetQuantity(int widgetId, int quantity)
{
    using(var conn = new SqlConnection("{connection string}")) {
        conn.Open();

        // create the transaction
        // You could use `var` instead of `SqlTransaction`
        using(SqlTransaction tran = conn.BeginTransaction()) {
            try
            {
                var sql = "update Widget set Quantity = @quantity where WidgetId = @id";
                var parameters = new { id = widgetId, quantity };

                // pass the transaction along to the Query, Execute, or the related Async
methods.
                conn.Execute(sql, parameters, tran);

                // if it was successful, commit the transaction
                tran.Commit();
            }
            catch(Exception ex)
            {
                // roll the transaction back
                tran.Rollback();

                // handle the error however you need to.
                throw;
            }
        }
    }
}
```

StackOverflow/。

。

```
// Widget has WidgetId, Name, and Quantity properties
public void InsertWidgets(IEnumerable<Widget> widgets)
{
    using(var conn = new SqlConnection("{connection string}")) {
        conn.Open();

        using(var tran = conn.BeginTransaction()) {
            try
            {
                var sql = "insert Widget (WidgetId,Name,Quantity) Values(@WidgetId, @Name,
@Quantity)";
                conn.Execute(sql, widgets, tran);
                tran.Commit();
            }
            catch(Exception ex)
            {
                tran.Rollback();
                // handle the error however you need to.
                throw;
            }
        }
    }
}
```

https://riptutorial.com/zh-TW/dapper/topic/6601/

4: DbGeographyDbGeometry

Examples

1. `Microsoft.SqlServer.Types`;Microsoft“Microsoft®SQLServer®2012Microsoft®SystemCLR” - x86x64。

2. `Dapper.EntityFramework` ;IDE“NuGet...”UI

```
install-package Dapper.EntityFramework
```

3. ;v11v10;`<configuration>app.configweb.config`

```
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="Microsoft.SqlServer.Types"
          publicKeyToken="89845dcd8080cc91" />
      <bindingRedirect oldVersion="10.0.0.0" newVersion="11.0.0.0" />
    </dependentAssembly>
  </assemblyBinding>
</runtime>
```

4. “dapper”

```
Dapper.EntityFramework.Handlers.Register();
```

```
string redmond = "POINT (122.1215 47.6740)";
DbGeography point = DbGeography.PointFromText(redmond,
    DbGeography.DefaultCoordinateSystemId);
DbGeography orig = point.Buffer(20); // create a circle around a point


var fromDb = connection.QuerySingle<DbGeography>(
  "declare @geos table(geo geography); insert @geos(geo) values(@val); select * from @geos",
  new { val = orig });

Console.WriteLine($"Original area: {orig.Area}");
Console.WriteLine($"From DB area: {fromDb.Area}");
```

DbGeographyDbGeometry https://riptutorial.com/zh-TW/dapper/topic/3984/dbgeography dbgeometry

5:

Examples

```
public async Task<Product> GetProductAsync(string productId)
{
    using (_db)
    {
        return await _db.QueryFirstOrDefaultAsync<Product>("usp_GetProduct", new { id =
productId },
            commandType: CommandType.StoredProcedure);
    }
}
```

```
public async Task SetProductInactiveAsync(int productId)
{
    using (IDbConnection con = new SqlConnection("myConnectionString"))
    {
        await con.ExecuteAsync("SetProductInactive", new { id = productId },
            commandType: CommandType.StoredProcedure);
    }
}
```

https://riptutorial.com/zh-TW/dapper/topic/1353/

6:

Examples

/◦ dapper`paramIDynamicParameters◦ AddParameters◦ DynamicParameters`

```
var p = new DynamicParameters(new { a = 1, b = 2 });
p.Add("c", dbType: DbType.Int32, direction: ParameterDirection.Output);
connection.Execute(@"set @c = @a + @b", p);
int updatedValue = p.Get<int>("@c");
```

-
-
-
-

RDBMS`Query`QueryMultiple` SQL ServerTDS◦

Dapper

```
connection.Execute(@"some Query with @a,@b,@c", new
{a=somevalueOfa,b=somevalueOfb,c=somevalueOfc});
```

```
public class SearchParameters {
  public string SearchString { get; set; }
  public int Page { get; set; }
}

var template= new SearchParameters {
  SearchString = "Dapper",
  Page = 1
};

var p = new DynamicParameters(template);
```

`Dictionary`

https://riptutorial.com/zh-TW/dapper/topic/12/

7:

this cnn	- this; - 。
<T> / Type	;/Type APIdynamicdynamicIDicionary<string,object> 。
sql	SQL
param	。
transaction	
buffered	IEnumerable
commandTimeout	;SqlMapper.Settings.CommandTimeout
commandType	;CommandText

RDBMS。 SQL Server@foo ; ?foo:foo。

Examples

SQL

DapperSQL。

dapper。 RDBMS@foo ?foo ?foo:foo dapper*foo*。

```
int id = 123;
string name = "abc";
connection.Execute("insert [KeyLookup](Id, Name) values(@id, @name)",
    new { id, name });
```

.....。 Dapper。

```
KeyLookup lookup = ... // some existing instance
connection.Execute("insert [KeyLookup](Id, Name) values(@Id, @Name)", lookup);
```

Dapper - `Description IsActive CreationDate` -

```
// TODO - removed for now; include the @Description in the insert
```

。

dapper/ - 。

```
connection.Execute("KeyLookupInsert", new { id, name },
    commandType: CommandType.StoredProcedure);
```

。 。 enum。

```
var orders = connection.Query<Order>(@"
select top (@count) * -- these brackets are an oddity of SQL Server
from Orders
where CustomerId = @customerId
and Status = @open", new { customerId, count = PageSize, open = OrderStatus.Open });
```

`customerId` - 。 RDBMSRDBMS。 Dapper - `{=name}@name` - 。 SQL。

```
var orders = connection.Query<Order>(@"
select top {=count} *
from Orders
where CustomerId = @customerId
and Status = {=open}", new { customerId, count = PageSize, open = OrderStatus.Open });
```

SQLDapperRDBMS

```
select top 10 *
from Orders
where CustomerId = @customerId
and Status = 3
```

RDBMS。 。 。

。

`IN (...)` 。 RDBMS - *RDBMS*。 dapper。 `IEnumerable`。 `@foo(@foo0,@foo1,@foo2,@foo3)` 4。 `IN`

```
int[] orderIds = ...
var orders = connection.Query<Order>(@"
select *
from Orders
where Id in @orderIds", new { orderIds });
```

SQL

```
select *
from Orders
where Id in (@orderIds0, @orderIds1, @orderIds2, @orderIds3)
```

`@orderIds0`arrray。 SQL。 SQL Server`OPTIMIZE FOR` / `UNKNOWN`;

```
option (optimize for
    (@orderIds unknown))
```

```
option (optimize for
    (@orderIds0 unknown, @orderIds1 unknown, @orderIds2 unknown, @orderIds3 unknown))
```

◦ `IEnumerable`Dapper`Execute`◦

```
Order[] orders = ...
// update the totals
connection.Execute("update Orders set Total=@Total where Id=@Id", orders);
```

dapper

```
Order[] orders = ...
// update the totals
foreach(Order order in orders) {
    connection.Execute("update Orders set Total=@Total where Id=@Id", order);
}
```

“”`async` API - dapper◦

```
await connection.ExecuteAsync(
    new CommandDefinition(
        "update Orders set Total=@Total where Id=@Id",
         orders, flags: CommandFlags.Pipelined))
```

◦

ADO.NETOleDB;`?`◦ Dapperdapper `?foo?` ;SQL`@foo:foo`dapper`?`◦

```
string region = "North";
int[] users = ...
var docs = conn.Query<Document>(@"
     select * from Documents
     where Region = ?region?
     and OwnerId in ?users?", new { region, users }).AsList();
```

`.region.users`SQL3

```
     select * from Documents
     where Region = ?
     and OwnerId in (?,?,?)
```

;。

```
declare @id int = ?id?; // now we can use @id multiple times in the SQL
```

-

```
int id = 42;
connection.Execute("... where ParentId = $id0$ ... SomethingElse = $id1$ ...",
      new { id0 = id, id1 = id });
```

https://riptutorial.com/zh-TW/dapper/topic/10/

8:

Examples

```
IDBConnection db = /* ... */
var id = /* ... */

db.Execute(@"update dbo.Dogs set Name = 'Beowoof' where Id = @id",
   new { id });
```

Dapper

```
var user = conn.Query<User>("spGetUser", new { Id = 1 },
                            commandType: CommandType.StoredProcedure)
           .SingleOrDefault();
```

```
var p = new DynamicParameters();
p.Add("@a", 11);
p.Add("@b",
      dbType: DbType.Int32,
      direction: ParameterDirection.Output);
p.Add("@c",
      dbType: DbType.Int32,
      direction: ParameterDirection.ReturnValue);

conn.Execute("spMagicProc", p,
             commandType: CommandType.StoredProcedure);

var b = p.Get<int>("@b");
var c = p.Get<int>("@c");
```

DataTableSQL Server。

```
CREATE TYPE [dbo].[myUDTT] AS TABLE([i1] [int] NOT NULL);
GO
CREATE PROCEDURE myProc(@data dbo.myUDTT readonly) AS
SELECT i1 FROM @data;
GO
/*
-- optionally grant permissions as needed, depending on the user you execute this with.
-- Especially the GRANT EXECUTE ON TYPE is often overlooked and can cause problems if omitted.
GRANT EXECUTE ON TYPE::[dbo].[myUDTT] TO [user];
GRANT EXECUTE ON dbo.myProc TO [user];
GO
*/
```

c

```
// Build a DataTable with one int column
DataTable data = new DataTable();
data.Columns.Add("i1", typeof(int));
// Add two rows
```

```
data.Rows.Add(1);
data.Rows.Add(2);

var q = conn.Query("myProc", new {data}, commandType: CommandType.StoredProcedure);
```

https://riptutorial.com/zh-TW/dapper/topic/5/

9:

- public static IEnumerable <T> Query <T>IDbConnection cnnsqlparam = nullSqlTransaction= nullbool buffered = true
- public static IEnumerable <dynamic> QueryIDbConnection cnnsqlparam = null SqlTransaction= nullbool buffered = true

	◦
SQL	◦
PARAM	◦
	◦
	◦ true◦ bufferedtrue`List<T> IEnumerable<T>`◦ bufferedfalsesql◦ ◦ buffered false ◦ sql◦

Examples

`Query<T>`◦

```
public class Dog
{
    public int? Age { get; set; }
    public Guid Id { get; set; }
    public string Name { get; set; }
    public float? Weight { get; set; }

    public int IgnoredProperty { get { return 1; } }
}

//
IDBConnection db = /* ... */;

var @params = new { age = 3 };
var sql = "SELECT * FROM dbo.Dogs WHERE Age = @age";

IEnumerable<Dog> dogs = db.Query<Dog>(sql, @params);
```

◦

```
IDBConnection db = /* ... */;
IEnumerable<dynamic> result = db.Query("SELECT 1 as A, 2 as B");

var first = result.First();
int a = (int)first.A; // 1
int b = (int)first.B; // 2
```

```
var color = "Black";
var age = 4;

var query = "Select * from Cats where Color = :Color and Age > :Age";
var dynamicParameters = new DynamicParameters();
dynamicParameters.Add("Color", color);
dynamicParameters.Add("Age", age);

using (var connection = new SqlConnection(/* Your Connection String Here */))
{
    IEnumerable<dynamic> results = connection.Query(query, dynamicParameters);
}
```

https://riptutorial.com/zh-TW/dapper/topic/3/

10:

- public static SqlMapper.GridReader QueryMultipleIDbConnection cnnsqlparam = null IDbTransaction= nullintcommandTimeout = nullCommandTypecommandType = null
- public static SqlMapper.GridReader QueryMultipleIDbConnection cnnCommandDefinition

SQL	sql
PARAM	
SqlMapper.GridReader	Dapper

Examples

QueryMultiple◦ GridReader◦

```
var sql = @"select * from Customers where CustomerId = @id
            select * from Orders where CustomerId = @id
            select * from Returns where CustomerId = @id";

using (var multi = connection.QueryMultiple(sql, new {id=selectedId}))
{
   var customer = multi.Read<Customer>().Single();
   var orders = multi.Read<Order>().ToList();
   var returns = multi.Read<Return>().ToList();
}
```

https://riptutorial.com/zh-TW/dapper/topic/8/

11:

`WriteToServer`、`WriteToServerAsync`、`IDataReader`、`DataTable`、`DataRow` 或 `DataRow[]` 。

Examples

`ToDataReader` 可以用來產生 `SqlBulkCopy` 的 `DataReader` 。

。

```
public class Widget
{
    public int WidgetId {get;set;}
    public string Name {get;set;}
    public int Quantity {get;set;}
}

public async Task AddWidgets(IEnumerable<Widget> widgets)
{
    using(var conn = new SqlConnection("{connection string}")) {
        await conn.OpenAsync();

        using(var bulkCopy = new SqlBulkCopy(conn)) {
            bulkCopy.BulkCopyTimeout = SqlTimeoutSeconds;
            bulkCopy.BatchSize = 500;
            bulkCopy.DestinationTableName = "Widgets";
            bulkCopy.EnableStreaming = true;

            using(var dataReader = widgets.ToDataReader())
            {
                await bulkCopy.WriteToServerAsync(dataReader);
            }
        }
    }
}
```

`ToDataReader` 可以用來產生 `SqlBulkCopy` 的 `DataReader` 。

。

```
public class Widget
{
    public int WidgetId {get;set;}
    public string Name {get;set;}
    public int Quantity {get;set;}
}

public void AddWidgets(IEnumerable<Widget> widgets)
{
    using(var conn = new SqlConnection("{connection string}")) {
        conn.Open();

        using(var bulkCopy = new SqlBulkCopy(conn)) {
            bulkCopy.BulkCopyTimeout = SqlTimeoutSeconds;
```

```
            bulkCopy.BatchSize = 500;
            bulkCopy.DestinationTableName = "Widgets";
            bulkCopy.EnableStreaming = true;

            using(var dataReader = widgets.ToDataReader())
            {
                bulkCopy.WriteToServer(dataReader);
            }
        }
    }
}
```

https://riptutorial.com/zh-TW/dapper/topic/6279/

12:

Examples

。

```
// Widget has WidgetId, Name, and Quantity properties
public async Task PurchaseWidgets(IEnumerable<Widget> widgets)
{
    using(var conn = new SqlConnection("{connection string}")) {
        await conn.OpenAsync();

        await conn.ExecuteAsync("CREATE TABLE #tmpWidget(WidgetId int, Quantity int)");

        // populate the temp table
        using(var bulkCopy = new SqlBulkCopy(conn)) {
            bulkCopy.BulkCopyTimeout = SqlTimeoutSeconds;
            bulkCopy.BatchSize = 500;
            bulkCopy.DestinationTableName = "#tmpWidget";
            bulkCopy.EnableStreaming = true;

            using(var dataReader = widgets.ToDataReader())
            {
                await bulkCopy.WriteToServerAsync(dataReader);
            }
        }

        await conn.ExecuteAsync(@"
            update w
            set Quantity = w.Quantity - tw.Quantity
            from Widgets w
                join #tmpWidget tw on w.WidgetId = tw.WidgetId");
    }
}
```

。 Dapper。 Dapper。

```
private async Task<IEnumerable<int>> SelectWidgetsError()
{
  using (var conn = new SqlConnection(connectionString))
  {
    await conn.ExecuteAsync(@"CREATE TABLE #tmpWidget(widgetId int);");

    // this will throw an error because the #tmpWidget table no longer exists
    await conn.ExecuteAsync(@"insert into #tmpWidget(WidgetId) VALUES (1);");

    return await conn.QueryAsync<int>(@"SELECT * FROM #tmpWidget;");
  }
}
```

```
private async Task<IEnumerable<int>> SelectWidgets()
{
  using (var conn = new SqlConnection(connectionString))
  {
```

```
    // Here, everything is done in one statement, therefore the temp table
    // always stays within the scope of the connection
    return await conn.QueryAsync<int>(
      @"CREATE TABLE #tmpWidget(widgetId int);
        insert into #tmpWidget(WidgetId) VALUES (1);
        SELECT * FROM #tmpWidget;");
  }
}

private async Task<IEnumerable<int>> SelectWidgetsII()
{
  using (var conn = new SqlConnection(connectionString))
  {
    // Here, everything is done in separate statements. To not loose the
    // connection scope, we have to explicitly open it
    await conn.OpenAsync();

    await conn.ExecuteAsync(@"CREATE TABLE #tmpWidget(widgetId int);");
    await conn.ExecuteAsync(@"insert into #tmpWidget(WidgetId) VALUES (1);");
    return await conn.QueryAsync<int>(@"SELECT * FROM #tmpWidget;");
  }
}
```

https://riptutorial.com/zh-TW/dapper/topic/6594/

13:

Examples

null vs DBNull

ADO.NET`null`◦ ;◦

- `nullDBNull.Value`
- `nullnull`

```
string name = null;
int id = 123;
connection.Execute("update Customer set Name=@name where Id=@id",
    new {id, name});
```

https://riptutorial.com/zh-TW/dapper/topic/13/

14:

.Net。

Examples

varcharIHtmlString

```
public class IHtmlStringTypeHandler : SqlMapper.TypeHandler<IHtmlString>
{
    public override void SetValue(
        IDbDataParameter parameter,
        IHtmlString value)
    {
        parameter.DbType = DbType.String;
        parameter.Value = value?.ToHtmlString();
    }

    public override IHtmlString Parse(object value)
    {
        return MvcHtmlString.Create(value?.ToString());
    }
}
```

TypeHandler

AddTypeHandlerSqlMapper。

```
SqlMapper.AddTypeHandler<IHtmlString>(new IHtmlStringTypeHandler());
```

```
SqlMapper.AddTypeHandler(new IHtmlStringTypeHandler());
```

Type

```
SqlMapper.AddTypeHandler(typeof(IHtmlString), new IHtmlStringTypeHandler());
```

https://riptutorial.com/zh-TW/dapper/topic/6/

S. No		Contributors
1	Dapper.NET	Adam Lear, balpha, Community, Eliza, Greg Bray, Jarrod Dixon, Kevin Montrose, Matt McCabe, Nick, Rob, Shog9
2	Multimapping	Devon Burriss
3		jhamm
4	DbGeography DbGeometry	Marc Gravell
5		Dean Ward, Matt McCabe, Nick, Woodchipper
6		Marc Gravell, Matt McCabe, Meer
7		4444, Marc Gravell, Nick Craver
8		Adam Lear, Jarrod Dixon, Sklivvz, takrl
9		Adam Lear, Chris Marisic, Cigano Morrison Mendez, Community, cubrr, Jarrod Dixon, jrummell, Kevin Montrose, Matt McCabe
10		Marc Gravell, Yaakov Ellis
11		jhamm
12		jhamm, Rob, takrl
13		Marc Gravell
14		Benjamin Hodgson, Community, Marc Gravell