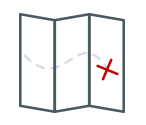
**Dapper** is a micro ORM or it is a simple object mapper framework which helps to map the native query output to a domain class or a C# class. It is a high performance data access system built by StackOverflow team and released as open source. ... Next, get the **Dapper** Nuget package installed into the project.

<https://dapper-tutorial.net/dapper>

Dapper FAQ

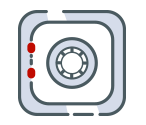
Is Dapper an ORM?

Yes and no! People are still arguing about it. Dapper has earned the title of king of the C# Micro ORM but is considered by multiple people as a simple object mapper for .NET.

Is Dapper better than Entity Framework?

Yes and no! People will prefer Dapper when they want to write the SQL query themselves with optimal performance.

Is Dapper SQL Injections safe?

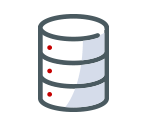
Yes, it's 100% safe if you use parametrized queries as you should always do!

Do Dapper support Bulk Insert?

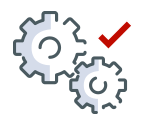
No, but a third-party library does: [Dapper Plus](https://dapper-plus.net/). It's a prime library that extend Dapper with all bulk operations.

[Learn More](https://dapper-plus.net/)

Do Dapper support my database provider?

Probably yes since Dapper provides extensions to the IDbConnection interface. It's your job to write the SQL compatible with your database provider.

Do Dapper support Transaction?

Yes, Dapper support transaction and every method that can use one have an optional parameter to specify it.

Sample code

IDbConnection db = \_dbService.GetDBConnection();

--------------------------------------

public SqlConnection GetDBConnection()

{

return \_context.Database.GetDbConnection() as SqlConnection;

}

--------------------------------------

public async Task<List<AuditDto>> AuditReport(int locationId, int employeeId, string eventType, System.DateTime fromDate, System.DateTime toDate, string connStrng)

{

IDbConnection db = \_dbService.GetDBConnection();

//using (IDbConnection db = new SqlConnection(connStrng))

{

return (await db.QueryAsync<AuditDto>("GetAuditReport",

new

{

LocationId = locationId,

EmployeeId = employeeId,

EventType = eventType,

FromDate = fromDate,

ToDate = toDate

},

commandType: CommandType.StoredProcedure))

.ToList();

}

}

{

return (await db.QueryAsync<DifferenceReportDto>(@"select

TBL\_Difference.Id,

Observation,

[Cashier].FirstName + ' ' + [Cashier].PaternalName as cashier,

[Supervisor].FirstName + ' ' + [Supervisor].PaternalName as supervisor

from

TBL\_Difference

inner join LKP\_Employee[Supervisor] on [Supervisor].Id = AuthorizerId

inner

join LKP\_Employee[Cashier] on [Cashier].Id = ReceiverId

inner

join TBL\_CashSelection on DifferenceId = TBL\_Difference.Id

inner

join TBL\_Withdrawal on TBL\_Withdrawal.Id = VerificationWithdrawalId

where TBL\_Difference.Id = @id",

new { id = differenceId },

commandType: CommandType.Text))

.FirstOrDefault();

}