

Capítulo 3

Dapper Extensión

Objetivo

Al finalizar el capítulo, el alumno:

- Comprender el uso de un micro ORM (Dapper).
- Realizar operaciones de acceso de datos y manipulación de objetos con Dapper.

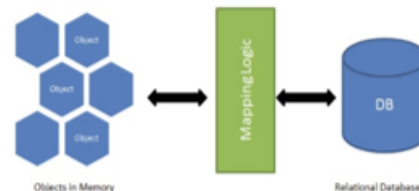
Temas

- Definición de Dapper
- Uso de Dapper
- Mapeo con Dapper
- Transacciones con Dapper.

1. Introducción a las transacciones

Definición de Dapper

- El un producto que permite mapeo objeto-relacional (ORM) para plataforma Microsoft .NET.
- Proporciona un marco para la asignación de un modelo de dominio orientado a objetos a bases de datos relacionales.
- Permite mitigar el tiempo invertido por parte del desarrollador en la programación de la persistencia de datos relacionales.
- Es un software de código abierto que se distribuye bajo una licencia dual:
 - Apache License 2.0
 - MIT



3 - 4

Copyright© Todos los Derechos Reservados - Cibertec Perú S.A.C.



¿Qué es un ORM? Un ORM (Object Relational Mapping) es un mapeo a un objeto relacional en nuestra aplicación, no es más que una técnica empleada para convertir los datos entre los distintos sistemas de base de datos y los lenguajes de programación. Creando una especie de base de datos virtual de objetos.

El mapeo objeto-relacional nos ayudará precisamente a olvidarnos completamente de cómo convertir los objetos en datos primitivos para almacenarlos y viceversa.

Si hablamos de ORM hay muchos en el mercado: el que se usará para este ejemplo será el Dapper.

En la página oficial se podrá ver más o menos en velocidad como es el dapper en comparación con otros tipos de mapeo. Dapper es un micro-ORM, ofreciendo los servicios principales de parametrización y materialización, pero por diseño, no toda la gama de servicios que usted esperaría en un completo ORM como LINQ to SQL o Entity Framework. En cambio, se centra en hacer la materialización lo más rápida posible, sin tantos gastos; sólo “ejecuta esta consulta y dame los datos”.

Lo podemos ver como un conjunto de clases que nos permiten mapear nuestras clases pocos con la base de datos.


Otra característica en especial del Dapper es que es independiente de la base de datos ya que este solo utiliza la interfaz `IDConnection`; y como la mayoría de bases de datos cuentan con una solo es implementar la correcta.

2. Uso con Dapper

Uso de Dapper

- Dapper hace la extensión de la interface `IDbConnection`.

```
public static IEnumerable<T> Query<T>(this  
IDbConnection cnn, string sql, object param =  
null, SqlTransaction transaction = null, bool  
buffered = true)
```

3 - 5Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.

Como podemos visualizar en el método implementado, la manera de utilizarlos es completamente sencillo y reduce la cantidad de líneas de código.

```
public Artist GetArtistById(int id)
{
    var storeName = "dbo.ArtistById";
    using (var connection = new SqlConnection(_connectionString))
    {
        var artist = connection.Query<Artist>(storeName, new { artistId=1 },
            commandType: CommandType.StoredProcedure
        ).SingleOrDefault();

        return artist;
    }
}
```

Para realizar operaciones dentro de una transacción, se debe configurar la propiedad `Connection` de un objeto `Command`, de tal forma, que haga referencia que está ejecutando la transacción, al objeto `Connection`; luego, se debe establecer el valor de la propiedad `Transaction` del objeto `Command` de tal forma, que haga referencia al objeto `Transaction`, el cual es generado por el `BeginTransaction`.

Si todas las sentencias SQL ejecutadas sobre una misma transacción se completan con éxito, entonces se debe hacer un llamado al método `Commit` del objeto `Transaction`. Si algún comando falla, entonces se debe hacer un llamado al método `RollBack` del objeto `Transaction`.

3. Transacciones distribuidas con System.Transaction

Transacciones con Dapper

- Al ser **Dapper** una extensión de ADO.Net las transacciones son similares:
 - System.Data.SqlClient.SqlTransaction
 - System.Data.OleDbClient.OleDbTransaction
 - System.Data.Odbc.OdbcTransaction



.NET Framework incluye el namespace System.Transactions, el cual proporciona soporte para transacciones distribuidas, que se ejecutan sobre administradores de transacciones relacionados a fuentes de datos o colas de mensajes (Message Queue).

De igual forma, la clase TransactionScope, la cual se encuentra dentro del namespace System.Transactions, permite crear y administrar transacciones distribuidas.

Para crear y ejecutar transacciones distribuidas, se debe instanciar un objeto de la clase TransactionScope y especificar si se desea crear un nuevo contexto de transacción o enlistar las transacciones en un contexto de transacción ya existente.