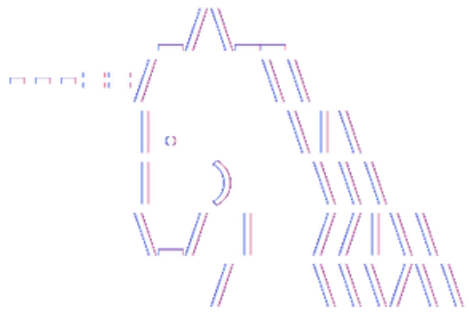

DbCommandInterceptor

INTERCEPT COMMAND WITH EFCORE

@romain-od

www.code-review.tech

READEREXECUTING

Called before executing a query command that returns a data reader.

```
public override InterceptionResult<DbDataReader> ReaderExecuting(
    DbCommand command,
    CommandEventData eventData,
    InterceptionResult<DbDataReader> result)
{
    // Perform logging or modification before executing a query command
    Console.WriteLine($"Executing query: {command.CommandText}");

    return base.ReaderExecuting(command, eventData, result);
}

public override ValueTask<InterceptionResult<DbDataReader>> ReaderExecutingAsync(
    DbCommand command,
    CommandEventData eventData,
    InterceptionResult<DbDataReader> result,
    CancellationToken cancellationToken = default)
{
    // Perform logging or modification before executing a query command asynchronously
    Console.WriteLine($"Executing query asynchronously: {command.CommandText}");

    return base.ReaderExecutingAsync(command, eventData, result, cancellationToken);
}
```

@romain-od

www.code-review.tech

NONQUERYEXECUTING

This method is called before executing a non-query command, such as insert, update, or delete.

```
public override InterceptionResult<int> NonQueryExecuting(
    DbCommand command,
    CommandEventData eventData,
    InterceptionResult<int> result)
{
    // Perform logging or modification before executing a non-query command
    Console.WriteLine($"Executing non-query: {command.CommandText}");

    return base.NonQueryExecuting(command, eventData, result);
}

public override ValueTask<InterceptionResult<int>> NonQueryExecutingAsync(
    DbCommand command, CommandEventData eventData,
    InterceptionResult<int> result,
    CancellationToken cancellationToken = default)
{
    // Perform logging or modification before executing a non-query command asynchronously
    Console.WriteLine($"Executing non-query asynchronously: {command.CommandText}");

    return base.NonQueryExecutingAsync(command, eventData, result, cancellationToken);
}
```

@romain-od

www.code-review.tech

SCALAREXECUTING

This method is called before executing a command that returns a single scalar value.

```
public override InterceptionResult<object> ScalarExecuting(
    DbCommand command,
    CommandEventData eventData,
    InterceptionResult<object> result)
{
    // Perform logging or modification before executing a scalar command
    Console.WriteLine($"Executing scalar: {command.CommandText}");

    return base.ScalarExecuting(command, eventData, result);
}

public override ValueTask<InterceptionResult<object>> ScalarExecutingAsync(
    DbCommand command,
    CommandEventData eventData,
    InterceptionResult<object> result,
    CancellationToken cancellationToken = default)
{
    // Perform logging or modification before executing a scalar command asynchronously
    Console.WriteLine($"Executing scalar asynchronously: {command.CommandText}");

    return base.ScalarExecutingAsync(command, eventData, result, cancellationToken);
}
```

@romain-od

www.code-review.tech

THANKS FOR READING

SHARE YOUR THOUGHTS

FOLLOW ME FOR MORE SHARING

@romain-od

www.code-review.tech