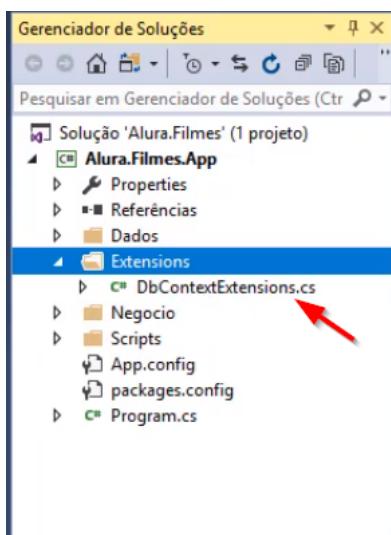


Configurando nomes de tabelas

Transcrição

Iremos quebrar a convenção do Entity que utiliza o nome da propriedade `DbSet` como nome de tabela, pois essa regra não é interessante para o projeto do nosso banco de dados legado. Para analisarmos de que forma Entity está gerando o SQL e como está tentando enviá-lo para o banco de dados, utilizaremos um método denominado `LogSQLToConsole`. Trata-se de um método de extensão, disponível na classe `DbContextExtensions.cs`. Basicamente, esse código é o mesmo que fizemos no **primeiro curso** para logar o SQL do Entity. Encapsulamos esse código em um método de extensão e podemos utilizá-lo neste momento.



```
using Alura.Filmes.App.Dados;
using Alura.Filmes.App.Extensions;

namespace Alura.Filmes.App
{
    class Program
    {
        static void Main(string[] args)
        {
            //select * from actor
            using (var contexto = new AluraFilmesContexto())
            {
                contexto.LogSQLToConsole();

                foreach (var ator in contexto.Atores)
                {
                    System.Console.WriteLine(ator);
                }
            }
        }
    }
}
```

Tentaremos iniciar a aplicação e novamente teremos o erro. Também será exibido o SQL que o Entity está tentando enviar para o banco de dados. O programa falhou ao executar o comando selecionado, e o problema está na "FROM [Atores] As [a]".

Teremos, portanto, de quebrar a convenção do Entity de utilizar a propriedade `DbSet` como nome de tabela.

```
Seletor C:\Windows\system32\cmd.exe
FROM [Atores] AS [a]

Failed executing DbCommand (24ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
SELECT [a].[Id], [a].[PrimeiroNome], [a].[UltimoNome]
FROM [Atores] AS [a] ←

Exceção Sem Tratamento: System.Data.SqlClient.SqlException: Invalid object name 'Atores'.
  em System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
  em System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)
  em System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)
  em System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady)
  em System.Data.SqlClient.SqlDataReader.TryConsumeMetaData()
  em System.Data.SqlClient.SqlDataReader.get_MetaData()
  em System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString, Boolean isInternal, Boolean forDescribeParameterEncryption)
  em System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolean async, Int32 timeout, Task& task, Boolean asyncWrite, Boolean inRetry, SqlDataReader ds, Boolean describeParameterEncryptionRequest)
  em System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method, TaskCompletionSource`1 completion, Int32 timeout, Task& task, Boolean& usedCache, Boolean asyncWrite, Boolean inRetry)
  em System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method)
  em System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String method)
```

Existem duas maneiras para quebrarmos a convenção do Entity. A primeira delas envolve decorarmos a classe `Ator` com uma anotação denominada `Table`. Nessa anotação, passaremos um argumento para o construtor da classe com o nome da tabela que deve ser mapeada. O nome da tabela, no caso, é `actor`.

```
using System.ComponentModel.DataAnnotations.Schema;

namespace Alura.Filmes.App.Negocio
{
    [Table]("actor")]
    public class Ator
    {
        public int Id { get; set; }
        public string PrimeiroNome { get; set; }
        public string UltimoNome { get; set; }
    }
}
```

Tentaremos executar a aplicação e receberemos novamente a mensagem de erro. Observaremos que o erro é de uma diferente natureza, dessa vez o nome da tabela (`actor`) está correto, como podemos ver. O erro está relacionado às colunas `Id`, `PrimeiroNome` e `UltimoNome`.

```
C:\Windows\system32\cmd.exe

Executing DbCommand [Parameters=[], CommandType='Text', CommandTimeout='30']
SELECT [a].[Id], [a].[PrimeiroNome], [a].[UltimoNome]
FROM [actor] AS [a]

Failed executing DbCommand (23ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
SELECT [a].[Id], [a].[PrimeiroNome], [a].[UltimoNome]
FROM [actor] AS [a]

Exceção Sem Tratamento: System.Data.SqlClient.SqlException: Invalid column name 'Id'.
Invalid column name 'PrimeiroNome'.
Invalid column name 'UltimoNome'.
   em System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapException)
   em System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapException)
   em System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)
   em System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataReader, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady)
   em System.Data.SqlClient.SqlDataReader.TryConsumeMetaData()
   em System.Data.SqlClient.SqlDataReader.get_MetaData()
   em System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String methodName, Boolean isInternal, Boolean forDescribeParameterEncryption)
   em System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, SqlDataReader dataReader, Boolean async, Int32 timeout, Task& task, Boolean asyncWrite, Boolean inRetry, SqlDataReader& result)

```

Ao analisarmos o `select` que foi montado, veremos que está usando o nome das colunas. Essa é a segunda convenção do Entity que aprendemos: o software utiliza o nome da propriedade que está na classe para mapear o nome da coluna na tabela. Teremos de quebrar essa regra.

```
C:\Windows\system32\cmd.exe

Executing DbCommand [Parameters=[], CommandType='Text', CommandTimeout='30']
SELECT [a].[Id], [a].[PrimeiroNome], [a].[UltimoNome]
FROM [actor] AS [a]

Failed executing DbCommand (23ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
SELECT [a].[Id], [a].[PrimeiroNome], [a].[UltimoNome]
FROM [actor] AS [a]

Exceção Sem Tratamento: System.Data.SqlClient.SqlException: Invalid column name 'Id'.
Invalid column name 'PrimeiroNome'.
Invalid column name 'UltimoNome'.
   em System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapException)
   em System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapException)
   em System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)
   em System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataReader, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady)
```