

1 - Linq to entities count

Transcrição

O novo pedido do cliente é realizar uma contagem do número total de faixas que existem para um determinado artista. Primeiro, vamos declarar um contexto:

```
using (var contexto = new AluraTunesEntities()) {  
}
```

Dentro disso, adicionaremos a variável `query` :

```
var query = from f in contexto.Faixas
```

Além do `select f` . Nesse momento não aplicaremos filtros:

```
using (var contexto = new AluraTunesEntities())  
{  
    var query = from f in contexto.Faixas  
                select f;  
}
```

Vamos acrescentar abaixo da variável `query` o `foreach()` :

```
foreach (var faixa in collection)
```

Depois disso, o `Console.WriteLine()` e o `ReadKey()` .

Ao executarmos a aplicação, teremos todas as faixas existentes no banco de dados. Então, precisaremos adicionar um filtro, assim, vamos inserir o

```
where f.Album.Artista.Nome == "Led Zeppelin"
```

Assim como o `ReadKey()` . Teremos:

```
using (var contexto = new AluraTunesEntities())  
{  
    var query = from f in contexto.Faixas  
                where f.Album.Artista.Nome == "Led Zeppelin"  
                select f;  
  
    foreach (var item in collection)  
    {  
        Console.WriteLine(faixa.Nome);  
    }  
}
```

```
}  
  
Console.ReadKey();
```

Ao executarmos o código, o resultado mostrado será todas as músicas do Led Zeppelin !

O interessante é utilizarmos todas as propriedades referentes à navegação: do `f` (de `faixa`) fomos até o `Album`, deste seguiremos para o `Artista` e apenas então chegaremos ao `Nome` - e por meio dele aplicaremos o filtro.

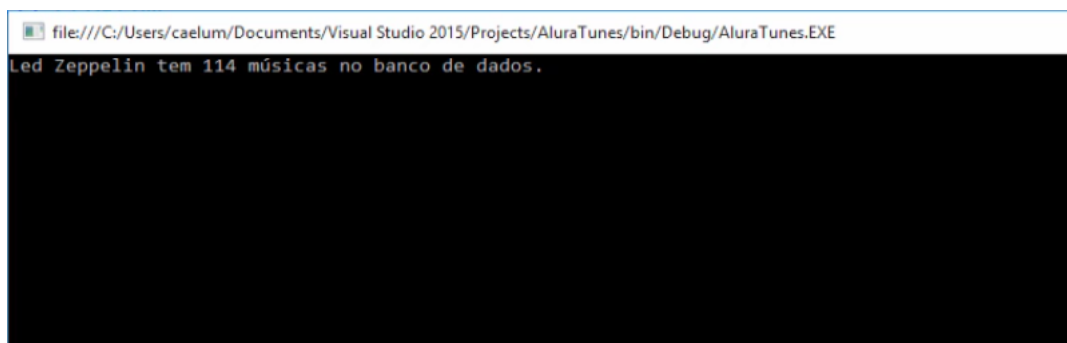
Nosso próximo objetivo é buscar a quantidade exata de músicas que existem relacionadas ao artista Led Zeppelin ! Vamos criar uma variável `quantidade` e junto disto adicionaremos o método `Count()`, que trará a contagem das músicas que estão na query. Portanto, vamos imprimir essa quantidade no console usando a linha:

```
Console.WriteLine("Led Zeppelin tem {0} músicas no banco de dados.", quantidade)
```

Depois, apagaremos o `foreach()` que não cabe mais:

```
using (var contexto = new AluraTunesEntities())  
{  
    var query = from f in contexto.Faixas  
                 where f.Album.Artista.Nome == "Led Zeppelin"  
                 select f;  
  
    var quantidade = query.Count();  
  
    Console.WriteLine("Led Zeppelin tem {0} músicas no banco de dados.", quantidade);  
  
}  
  
Console.ReadKey();  
}
```

Ao executarmos temos:



O resultado mostrará 114 músicas no total!

Outra maneira de construir a mesma consulta seria utilizar a sintaxe de método, pois, é por meio dela que pegaremos os dados utilizando o contexto. Vamos montar essa nova consulta utilizando sintaxe de método, assim, primeiro declaramos a variável `quantidade` e iremos igualá-la com `contexto.Faixas`. A partir do `faixa` já poderemos utilizar o método `Count`, assim teremos a seguinte linha:

```
var quantidade = contexto.Faixas.Count();
```

Depois, adicionaremos:

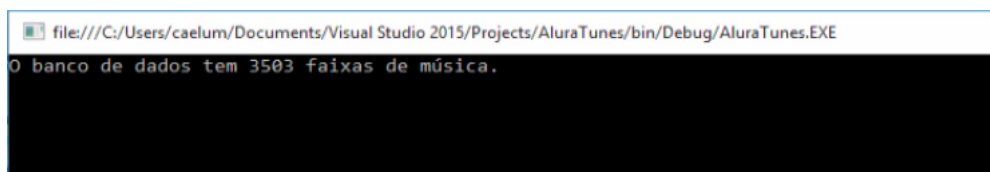
```
Console.WriteLine("O banco de dados tem {0} faixas de música.", quantidade)
```

Note que não utilizamos o `from` ! o trecho do código ficará assim:

```
var quantidade = contexto.Faixas.Count();

Console.WriteLine("O banco de dados tem {0} faixas de música.", quantidade);
```

Ao executarmos o código, o resultado será diferente:

A screenshot of a Windows console window. The title bar shows the file path: file:///C:/Users/caelum/Documents/Visual Studio 2015/Projects/AluraTunes/bin/Debug/AluraTunes.EXE. The console output is: "O banco de dados tem 3503 faixas de música."

Porém, nosso objetivo não é pegar todas as faixas existentes, mas sim aplicar um filtro. Assim, dentro de `var quantidade` vamos inserir uma expressão `lambda` que vai representar o filtro. Adicionaremos `.Count()` :

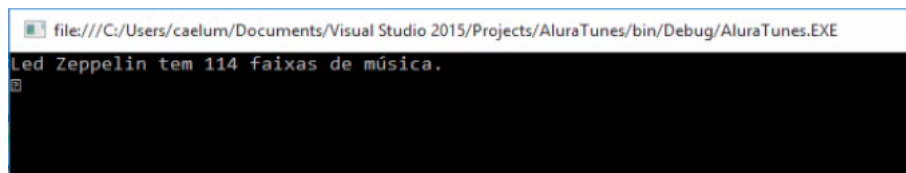
```
.Count(f => f.Album.Artista.Nome == "Led Zeppelin")
```

Com isto, é preciso modificar o `Console.WriteLine()` e alterar texto para `"Led Zeppelin tem {0} faixas de música."`. Teremos:

```
var quantidade = contexto.Faixas
    .Count(f => f.Album.Artista.Nome == "Led Zeppelin");

Console.WriteLine("O banco de dados tem {0} faixas de música.", quantidade);
```

Por fim, conseguimos o mesmo resultado da primeira consulta!

A screenshot of a Windows console window. The title bar shows the file path: file:///C:/Users/caelum/Documents/Visual Studio 2015/Projects/AluraTunes/bin/Debug/AluraTunes.EXE. The console output is: "Led Zeppelin tem 114 faixas de música."

O que fizemos foi simplesmente trocar uma sintaxe de consulta por uma de método!

```
using (var contexto = new AluraEntities())
{
    var query = from f in contexto.Faixas
                 where f.Album.Artista.Nome == "Led Zeppelin"
                 select f;

    //var quantidade = query.Count();

    //Console.WriteLine("Led Zeppelin tem {0} músicas no banco de dados.",
    //                    quantidade);

    var quantidade = contexto.Faixas
                           .Count(f => f.Album.Artista.Nome == "Led Zeppelin");

    Console.WriteLine("Led Zeppelin tem {0} faixas de música.", quantidade)
}
```

Nesta aula, vimos como fazer uma contagem de duas maneiras distintas!