03

2 - Linq com joins

Transcrição

Nesta aula, vamos começar a criar consultas que combinam dados de origens diferentes! Por enquanto não existe uma classe que armazene as músicas, por isso, é preciso criá-la. Adicionamos `class Musica` e dentro disso acrescentamos `Id` :

```
public int Id { get; set; }
```

Além disso, também:

```
public string Nome { get; set; }
```

Ainda dentro da classe `Musica` , é preciso inserir a informação sobre gênero, portanto, acrescentamos `public int GeneroId { get; set; }` e teremos:

```
class Musica
{
    public int Id { get; set; }
    public string Nome { get; set; }
    public int GeneroId { get; set; }
}
```

Para cadastrar as músicas, criamos uma variável que deve conter uma lista de músicas, a `var musicas = new List<Music>` . A lista estará vazia inicialmente, então, começaremos a cadastrar as músicas. Colocaremos esse código logo abaixo do `Console.WriteLine("{0}t{1}", genero.Id, genero.home)` . Teremos:

```
//listar músicas

var musicas = new List<Musica>
{
    new Musica { Id = 1, Nome = "Sweet Child O'Mine", GeneroId = 1 },
    new Musica { Id = 2, Nome = "I Shoot The Sheriff", GeneroId = 2},
    new Musica { Id = 3, Nome = "Danúbio Azul", GeneroId = 5},
}
```

Com isso a lista de músicas será cadastrada!

Agora, vamos criar uma consulta que pegue os dados da lista e para isso utilizaremos a linguagem `LINQ` que já aprendemos a manusear:

```
var musicaQuery = from m in musicas
                  select m;
```

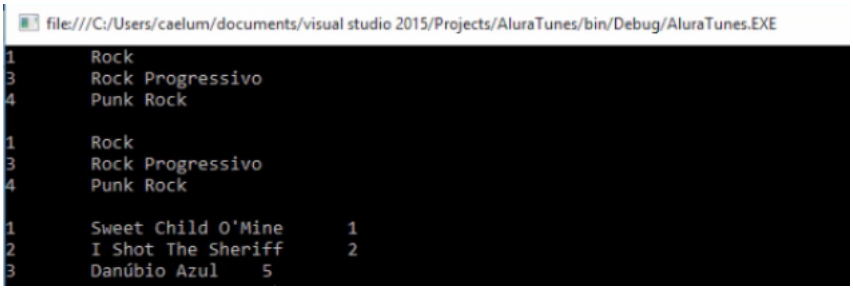
É preciso também imprimir as músicas, assim, usaremos o `foreach(var musica in musicaQuery)` e o `Console.WriteLine()`. Dentro deste último, colocaremos o formato e `musica.Id, musica.Nome, musica.GeneroId` e os indíces `"{0}\t{1}\t{2}"`. Teremos:

```
var musicaQuery = from m in musicas
                  select m;

foreach(var musica in musicaQuery)
{
    Console.WriteLine(`"{0}\t{1}\t{2}", musica.Id, musica.Nome, musica.GeneroId);
}
```

Acrescentar uma linha entre as duas listagens para diferenciá-las! E adicionamos mais um `Console.WriteLine()`

```
var musicaQuery = from m in musicas
                  select m;


console.WriteLine();
foreach(var musica in musicaQuery)
{
    Console.WriteLine("{0}\t{1}\t{2}", musica.Id, musica.Nome, musica.GeneroId);
}
```

E rodando isto temos o seguinte:



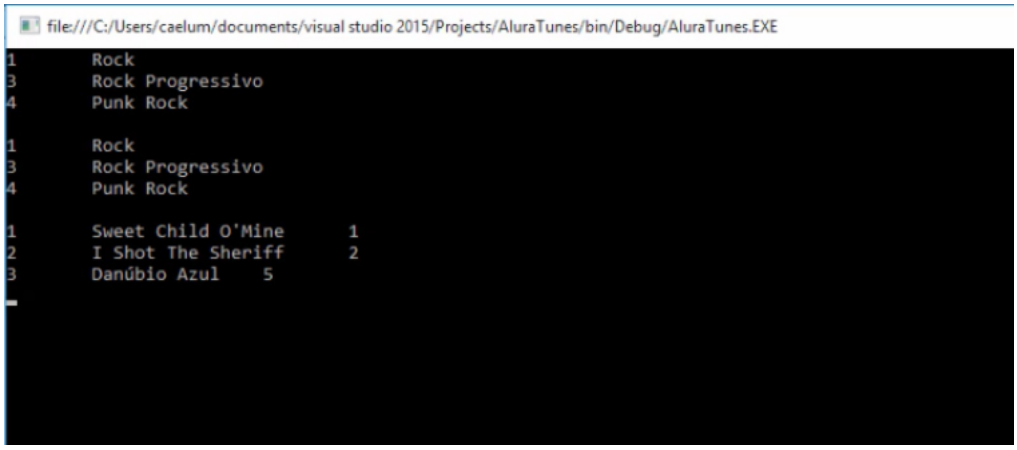
Entretanto, note que a listagem ainda não está bacana, pois o `Id` do gênero segue sem informar nada! Por isso, vamos modificar a consulta para incluir também a listagem de gêneros!

O problema é: **Como é possível fazer uma consulta que junte duas entidades diferentes?**

Para resolver esta situação basta utilizar uma claúsula `inner join` e para isso é preciso relacionar duas tabelas. Em nosso caso, no código `C#`, não existem duas tabelas, portanto, é preciso trabalhar com objetos do sistema. Dessa maneira, nós vamos fazer um `join` utilizando a listagem de música. Assim, escrevemos `join g in genero` e utilizando o `on`, colocaremos as duas propriedades que se relacionam. No caso, o objeto de tipo `gênero` e o objeto de tipo `música`. Adicionaríamos `m.GeneroId == g.Id` se a sintaxe do `join` permitisse o `==`, no entanto, somos obrigados a substituir por `equals`:

```
var musicQuery = from m in musicas
                 join g in genero on m.GeneroId equals g.Id
                 select m;
```

Rodando a consulta temos:



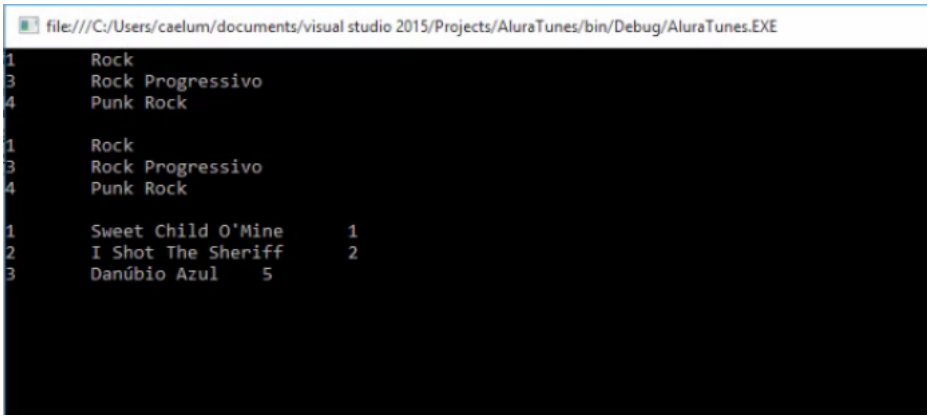
Mas, repare que isso não muda absolutamente nada na aplicação! Não basta trazer apenas o `m` é preciso também pegar o `g`. Para fazer isso vamos utilizar um objeto anônimo. Trata-se do objeto que não possui nome e já está pronto para ser aplicado. Por isso, iremos escrever `select new {}` e passaremos as propriedades `m` e `g`:

```
var musicQuery = from m in musicas
                 join g in genero on m.GeneroId equals g.Id
                 select new { m, g};
```

Porém, na linha do `Console.WriteLine` será acusado um erro na sintaxe. Isso ocorre, pois não existe mais o `musica.ID`, `musica.Nome` e `musica.genero Id`, o que temos é um objeto com propriedade `m` e `g`. Assim, não é preciso acessar essas propriedades diretamente, portanto, basta escrever `musica.m.Id`, `musica.m.Nome` e `musica.m.GeneroId`:

```
foreach(var musica in musicaQuery)
{
    Console.WriteLine("{0}\t{1}\t{2}", musica.m.Id, musica.m.Nome, musica.m.GeneroId);
}
```

E, rodando a aplicação teremos:

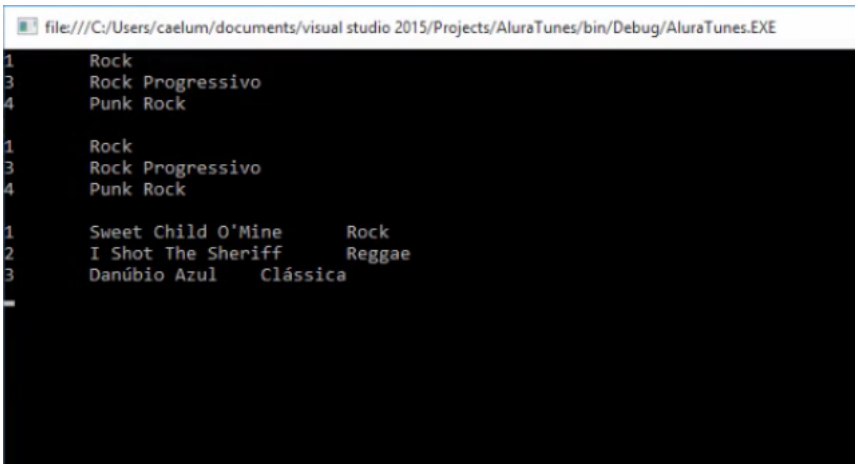


Ou seja, nada de novo ocorre! Falta inserir o nome do gênero. Assim, modificaremos `musica.m.GeneroId` por `musica.g.Nome` que equivale ao nome do gênero:

```
foreach(var musica in musicaQuery)
{
    Console.WriteLine("{0}\t{1}\t{2}", musica.m.Id, musica.m.Nome, musica.g.Nome);
```

```
}
```

E ao rodar novamente teremos:



Assim, conseguimos finalizar a primeira consulta utilizando como recurso o `LINQ` !

Imagina se fizéssemos a mesma consulta, mas utilizando laços? Seria bem mais complicado!