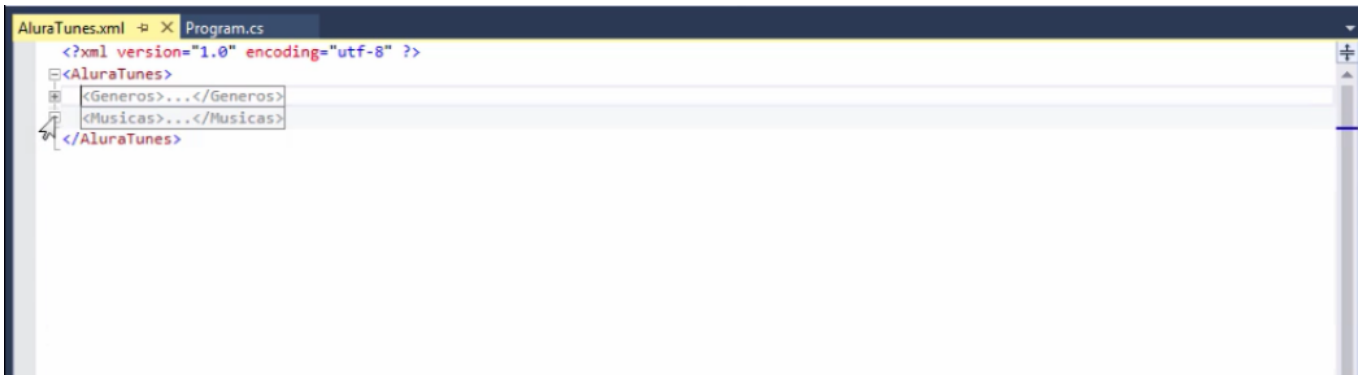
01

1 - Linq to Xml

Transcrição

Já vimos como utilizar o `LINQ` para acessar dados em memória. Agora, vamos explorar outras oportunidades disponíveis no `LINQ`. Por exemplo, acessar dados `xml`.

O cliente enviou um arquivo `.xml` para nós e ele está aberto:



Esse arquivo é muito simples, e contem gêneros e músicas. Ou seja, ele é apenas um sub conjunto do que seria um banco de dados! Assim, vamos começar a acessar os dados do arquivo `.xml` para montar uma consulta e no final teremos um resultado muito semelhante ao `Object`.

Para acessar o `xml` é preciso utilizar uma biblioteca própria do `.NET`. Então, primeiro, instancia-se uma variável que vai representar a raiz do arquivo `.xml` e também declara-se um `XElements` de nome `root`: `XElement root = XElement.Load()`. Passamos o arquivo que o cliente mandou e está na pasta `"Data\AluraTunes.xml"`. Com isso, estamos acessando o elemento raiz do arquivo. O próximo passo será definir uma consulta `LINQ` que acesse os dados `xml`. Para isto, vamos declarar a variável `queryXML` e como devemos puxar os dados do arquivo, utilizaremos a sintaxe do `LINQ`, visto anteriormente. Portanto, usamos `from g In root` e começamos a acessar as coleções de dados do arquivo `xml`. Vamos escrever `root.Element("")` e no arquivo `AluraTunes.xml` observamos a existência do nó `Generos` e passamos a seguinte informação:

```
class Program
{
    static void Main(string[] args)
    {
       XElement root = XElement.Load(@"Data\AluraTunes.xml");

        var querySQL =
            from g In root.Element("Generos")

    }
}
```

Ainda, acrescentaresmos `from g In root.Element("Generos")` e `Elements("Genero")`. Se o `xml` contem apenas texto, como podemos acessá-lo por meio do `LINQ`? Porém, este não sabe acessar o `xml` diretamente. Nós o que estamos fazendo é acessar o `xml` por meio de uma biblioteca `.NET` própria para `.xml`. O `system.xml.linq` é essa biblioteca, especializada em trazer para o `LINQ` todos os objetos e elementos que fazem parte do `.xml`. O que vamos fazer é trazer elementos do `Genero` e jogar na consulta e também vamos retornar elementos da consulta que estão no objeto `g` usando o `select g`. Falta pedir

para imprimir informações no Console, assim, utilizaremos o metódo `foreach()` para fazer uma varredura dos gêneros, portanto, escrevemos `foreach (var genero In queryXML)`.

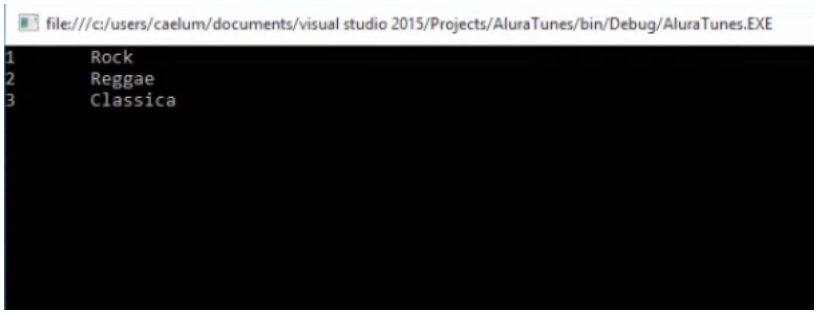
Também adicionaremos `Console.WriteLine()` que servirá para imprimir e desejamos passar para ele o `generoId` que só poderá ser acessado usando o objeto `genero.element` e dentro dessa string colocaremos o `"GeneroId"`. Além disso, vamos adicionar também a propriedade `Value`. Já adicionamos o `GeneroId`, em seguida, vamos incluir o nome do `Genero`:

```
class Program
{
    static void Main(string[] args)
    {
       XElement root = XElement.Load(@"Data\AluraTunes.xml");

        var querySQL =
            from g In root.Element("Generos").Elements("Genero");
            select g;

        foreach (var genero In queryXML)
        {
            Console.WriteLine(genero.element("GeneroId")).Value);
        }
}
```

Vamos incluir também o elemento do nome do gênero, portanto, escrevemos `genero.Element("Nome").Value`. Falta apenas formatar a saída para o `Console`, para isso adicionamos uma `string` de formatação: `"{0}\t{1}"`. Falta acrescentar o `Console.ReadKey()` e teremos:

```
foreach (var genero In queryXML)
        {
            Console.WriteLine(genero.element("GeneroId").Value, genero.Element("Nome").Value);


        }
        Console.ReadKey();
```

Rodando temos o seguinte:



Agora, vamos combinar os dados de música e gênero e trazer ambos em apenas uma consulta! Assim, vamos utilizar o `join`, portanto, criamos uma consulta `query` e trazemos o elemento `g` que está no `root.Elements("Generos")`. Depois, incluiremos o `Elements("Genero")`:

```
var query = from g in root.Element("Generos").Elements("Genero")
```

Falta acrescentar o `join`, feito com as músicas:

```
join m in root.Element("Musicas").Elements("Musica")
```

Agora, resta dizer para a cláusula `join` as propriedades dos elementos `genero` e `musica`, dessa maneira, adicionamos `on g.Element("GeneroId").Value` e isso deverá ser igual, `equals`, a `m.Element(GeneroId).Value`. Vamos acrescentar, ainda, a cláusula `select new` para trazer os dados. Passamos as propriedades que serão trazidas na consulta, no caso `Musica = m.Element "Nome").Value`.

Como falta o `genero`, adicionaremos:

```
Genero = g.Element("Nome").Value
```

Feito isto, vamos imprimir o resultado da listagem:

```
var query = from g in root.Element("Generos").Elements("Genero")
            join m in root.Element("Musicas").Elements("Musica")
                on g.Element("GeneroId").Value equals m.Element("GeneroId").Value
            select new { Musica = m.Element("Nome").Value, Genero = g.Element("Nome").Va

foreach (var item in query)
{
    Console.WriteLine("{0}\t{1}", item.Musica, item.Genero);

}
```

Ainda falta a impressão do resultado da listagem, assim, escrevemos mais um `Console.WriteLine()` e `foreach (var musicaEgenero in query)`. Repare que o nome da variável é `musicaEgenero`. Assim, estamos dizendo que para cada `MusicaEgenero` e vamos trazer os dados da `query`. Ainda, vamos imprimir a informação usando a seguinte linha:

```
Console.WriteLine(musicaEgenero.Musica, musicaEgenero.Genero)
```

Junto disso, falta colocarmos também a formatação no `Console`, o `{0}\t{1}`:

```
Console.WriteLine();

foreach (var musicaEgenero in query)
{
    Console.WriteLine("{0}\t{1}", musicaEgenero.Musica, musicaEgenero.Genero)'
}
```

E rodando temos o seguinte resultado:

Acabamos de ver o `LINQ` do `xml` e não é diferente do `Object`