

First(), Last() and Single() Methods Lab

Lab 1: FirstOrDefault

Open the **Program.cs** file from your last lab.

Replace all the code in this file to look like the following:

```
using LINQLab.EntityClasses;
using LINQLab.RepositoryClasses;

// Declare variables and fill data
List<Song> songs = SongRepository.GetAll();
Song? value = null;

// TODO: Write your Query Here

// Was the data in the collection?
if (value == null) {
    Console.WriteLine("Genre Not Found");
}
else {
    // Display the Data Found
    Console.WriteLine(value);
}

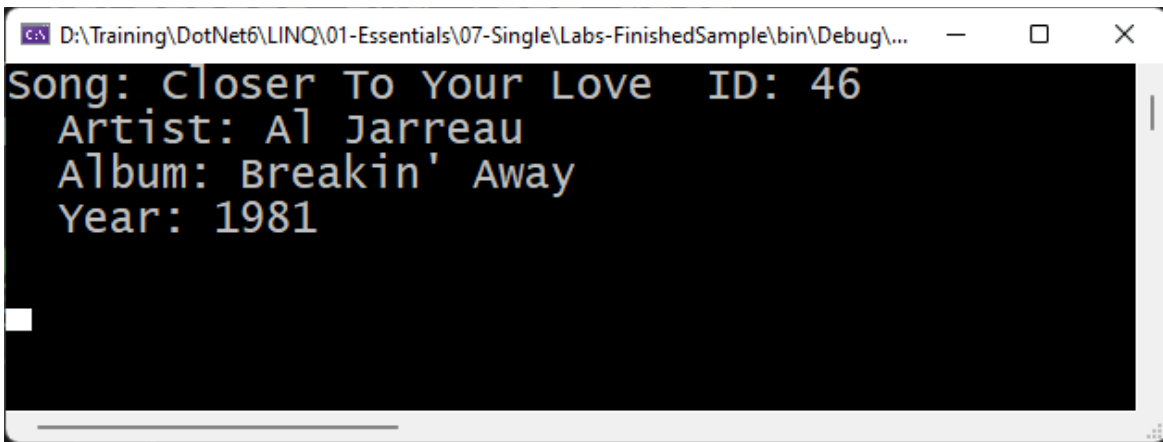
// Pause for Results
Console.ReadKey();
```

Write a query to select the *first* song with a **GenreId = 17**.

Use the **FirstOrDefault()** method for this query.

Try it Out

Run the application and you should see the following results.



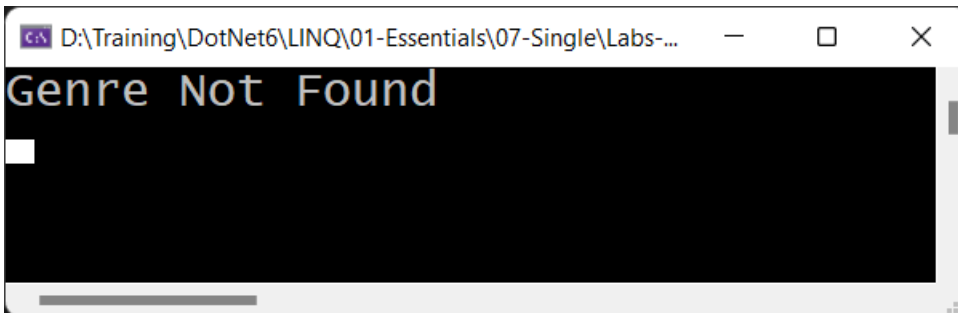
```

D:\Training\DotNet6\LINQ\01-Essentials\07-Single\Labs-FinishedSample\bin\Debug\...
Song: Closer To Your Love ID: 46
Artist: Al Jarreau
Album: Breakin' Away
Year: 1981

```

Try Out Invalid Genre

Next, try an invalid **GenreId** such as **77** to see the error message appear.



```

D:\Training\DotNet6\LINQ\01-Essentials\07-Single\Labs-...
Genre Not Found

```

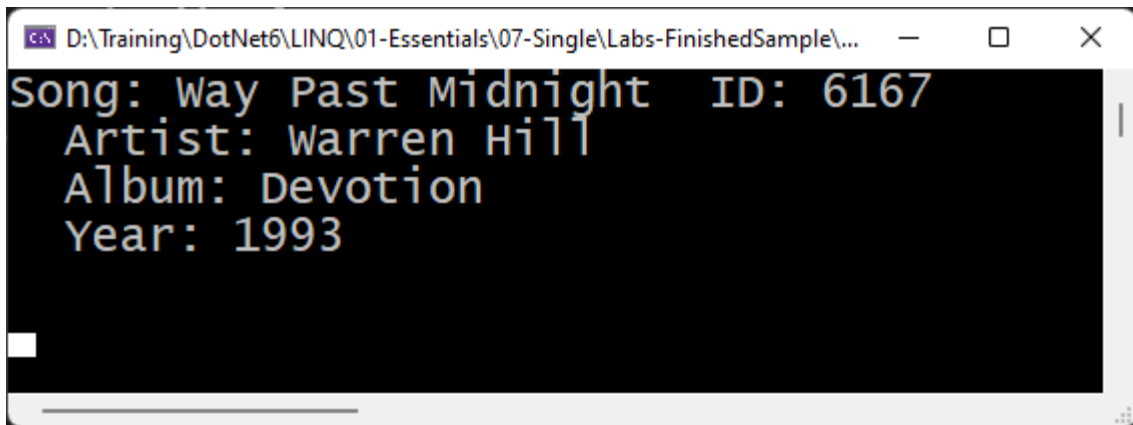
Lab 2: LastOrDefault

Write a query to select the *last* song with a **GenreId** = 17.

Use the LastOrDefault() method for this query.

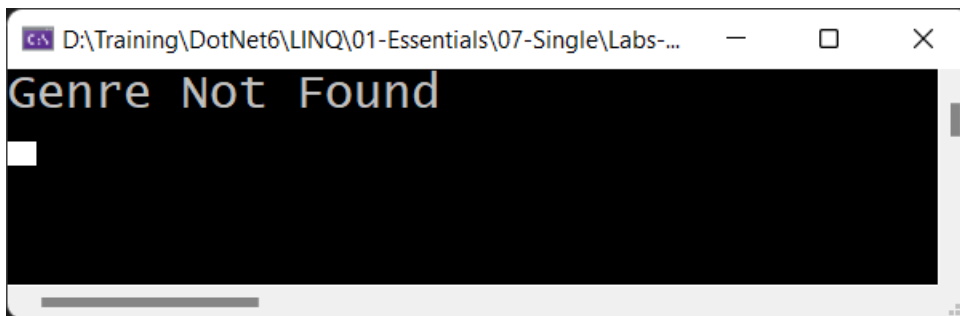
Try it Out

Run the application and you should see the following results.



Try Out Invalid Genre

Next, try an invalid **GenreId** such as **77** to see the error message appear.



Lab 3: FirstOrDefault with Default

Change the last query back to use the FirstOrDefault() method.

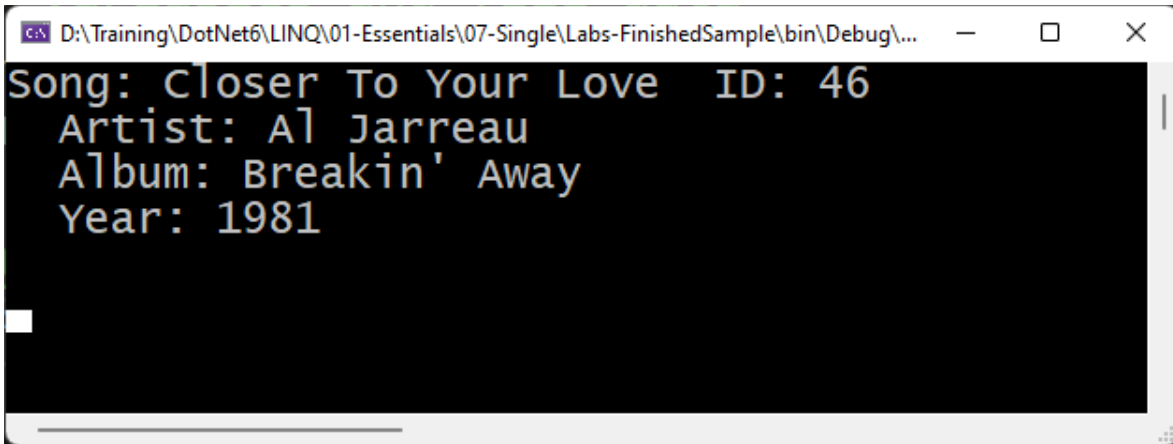
Change the **GenreId** to search for to **17**.

Add the second parameter to the FirstOrDefault() method call to return a new **Song** object with the following properties set.

```
SongId = -1  
SongName = "Genre Not Found"
```

Try it Out

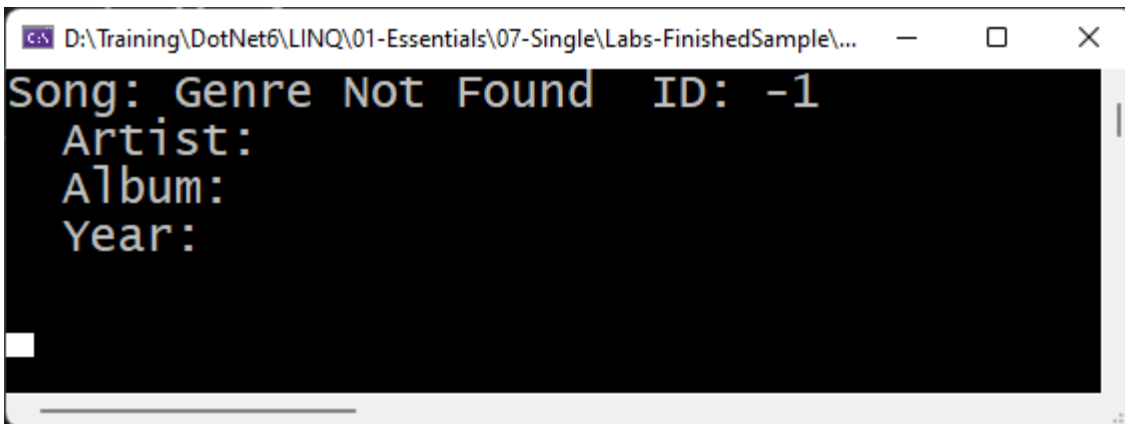
Run the application and you should see the following results.



```
C:\> D:\Training\DotNet6\LINQ\01-Essentials\07-Single\Labs-FinishedSample\bin\Debug\...  
Song: Closer To Your Love ID: 46  
Artist: Al Jarreau  
Album: Breakin' Away  
Year: 1981
```

Try Out Invalid Genre

Next, try an invalid **GenreId** such as **77** to see the following result.



```
C:\> D:\Training\DotNet6\LINQ\01-Essentials\07-Single\Labs-FinishedSample\...  
Song: Genre Not Found ID: -1  
Artist:  
Album:  
Year:
```

Lab 4: LastOrDefault with Default

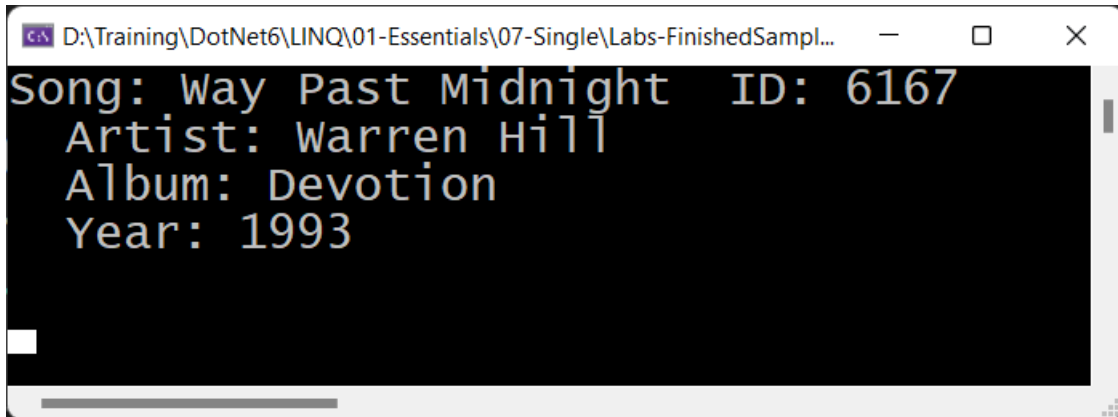
Change the last query back to use the LastOrDefault() method.

Change the **GenreId** to the value **17**.

Keep the second parameter to the LastOrDefault() method just like in the previous lab.

Try it Out

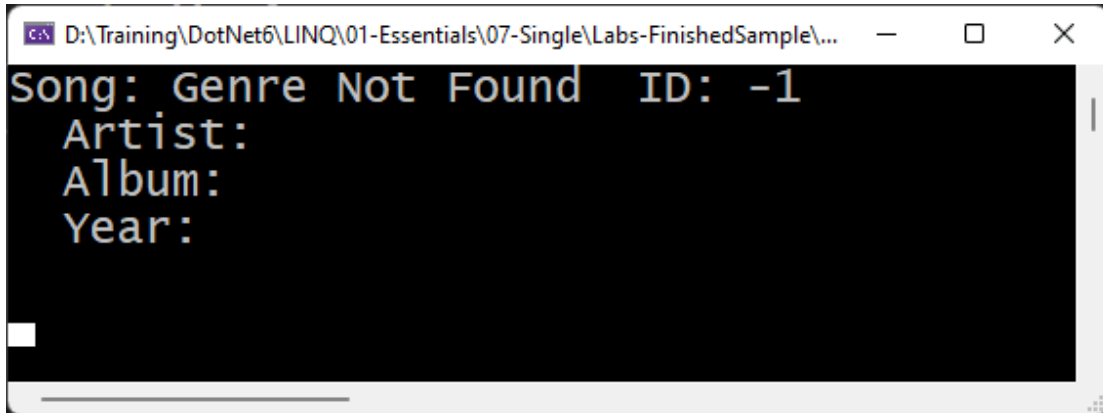
Run the application and you should see the following results.



```
D:\Training\DotNet6\LINQ\01-Essentials\07-Single\Labs-FinishedSample\...  
Song: Way Past Midnight ID: 6167  
Artist: Warren Hill  
Album: Devotion  
Year: 1993
```

Try Out Invalid Genre

Next, try an invalid **GenreId** such as **77** to see the following result.



```
D:\Training\DotNet6\LINQ\01-Essentials\07-Single\Labs-FinishedSample\...  
Song: Genre Not Found ID: -1  
Artist:  
Album:  
Year:
```

Lab 5: First

Open the **Program.cs** file and make it look like the following:

```
using LINQLab.EntityClasses;
using LINQLab.RepositoryClasses;

// Declare variables and fill data
List<Song> songs = SongRepository.GetAll();
Song? value = null;

try {
    // TODO: Write your Query Here

    // Display the Data Found
    Console.WriteLine(value);
}
catch (InvalidOperationException) {
    Console.WriteLine("Genre Not Found");
}
catch (Exception ex) {
    Console.WriteLine(ex.Message);
}

// Pause for Results
Console.ReadKey();
```

Write a query using the First() method to locate the first **Song** with a **GenreId** = 17.

Try it Out

Run the application and you should see the results you saw in Lab 1.

Try Out Invalid Genre

Next, try an invalid **GenreId** such as 77

Set a breakpoint on the line with your query.

Run the application and watch the code jump to the catch block.

Lab 6: Last

Change the method in the previous lab to use Last() instead of First().

Try it Out

Run the application and you should see the results you saw in Lab 2.

Try Out Invalid Genre

Next, try an invalid **GenreId** such as **77** to see the error message appear.

Lab 7: Single

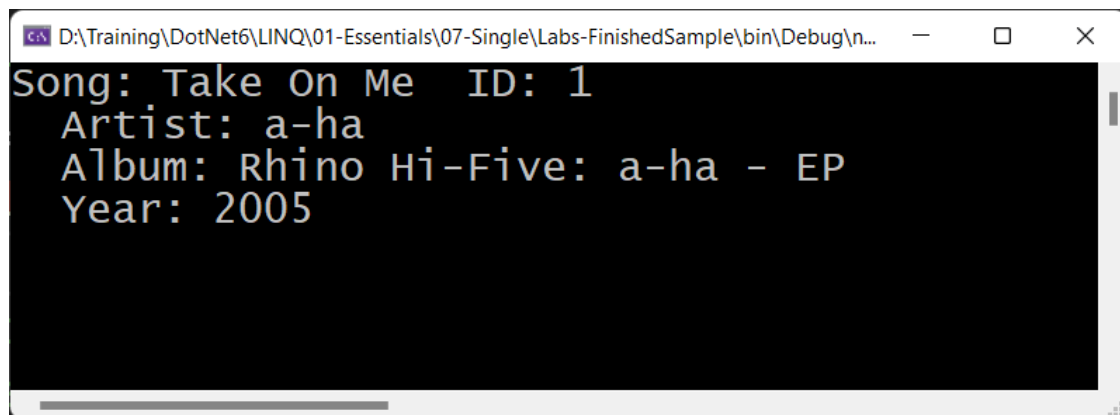
Modify the `InvalidOperationException` block to display the value of the **Message** property from the **Exception** object.

```
catch (InvalidOperationException ex) {  
    // Not Found or Multiple Items Found  
    Console.WriteLine(ex.Message);  
}
```

Replace the query you wrote in the last lab to select just a single **Song** where the **SongId** property is equal to the value **1**.

Try it Out

Run the application and you should see the following results.

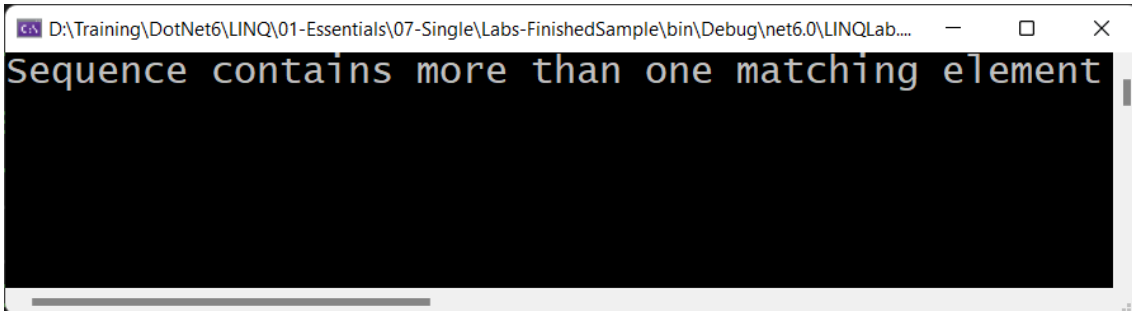


Multiple Values Found

Replace the query you just wrote to select just a single **Song** where the **GenreId** property is equal to **17**.

```
value = (from row in songs select row)
        .Single(row => row.GenreId == 17);
```

This causes the **Message** property to be displayed from the Exception object.



No Values Found

Modify the code to select a **GenreId** such as **77**.

```
value = (from row in songs select row)
        .Single(row => row.GenreId == 77);
```

This causes the **Message** property to be displayed from the Exception object.

