LINQ Query Expressions (C# Programming Guide)

Visual Studio 2015
Other Versions

-

Language-Integrated Query (LINQ) is the name for a set of technologies based on the integration of query capabilities directly into the C# language (also in Visual Basic and potentially any other .NET language). With LINQ, a query is now a first-class language construct, just like classes, methods, events and so on.

For a developer who writes queries, the most visible "language-integrated" part of LINQ is the query expression. Query expressions are written in a declarative *query syntax* introduced in C# 3.0. By using query syntax, you can perform even complex filtering, ordering, and grouping operations on data sources with a minimum of code. You use the same basic query expression patterns to query and transform data in SQL databases, ADO.NET Datasets, XML documents and streams, and .NET collections.

The following example shows the complete query operation. The complete operation includes creating a data source, defining the query expression, and executing the query in a **foreach** statement.

C#
Copy

```
class LINQQueryExpressions
{
    static void Main()
    {

        // Specify the data source.
        int[] scores = new int[] { 97, 92, 81, 60 };

        // Define the query expression.
        IEnumerable<int> scoreQuery =
            from score in scores
            where score > 80
```

```
            select score;

        // Execute the query.
        foreach (int i in scoreQuery)
        {
            Console.Write(i + " ");
        }
    }
}
// Output: 97 92 81
```

For more information about the basics of LINQ in C#, see Getting Started with LINQ in C#.

Query Expression Overview

- Query expressions can be used to query and to transform data from any LINQ-enabled data source. For example, a single query can retrieve data from a SQL database, and produce an XML stream as output.
- Query expressions are easy to master because they use many familiar C# language constructs. For more information, see Getting Started with LINQ in C#.
- The variables in a query expression are all strongly typed, although in many cases you do not have to provide the type explicitly because the compiler can infer it. For more information, see Type Relationships in LINQ Query Operations (C#).
- A query is not executed until you iterate over the query variable in a **foreach** statement. For more information, see Introduction to LINQ Queries (C#).
- At compile time, query expressions are converted to Standard Query Operator method calls according to the rules set forth in the C# specification. Any query that can be expressed by using query syntax can also be expressed by using method syntax. However, in most cases query syntax is more readable and concise. For more information, see C# Language Specification and Standard Query Operators Overview.
- As a rule when you write LINQ queries, we recommend that you use query syntax whenever possible and method syntax whenever necessary. There is no semantic or performance difference between the two different forms. Query expressions are often more readable than equivalent expressions written in method syntax.
- Some query operations, such as Count<TSource> or Max, have no equivalent query expression clause and must therefore be expressed as a method call. Method syntax can be combined with query syntax in various ways. For more information, see Query Syntax and Method Syntax in LINQ (C#).
- Query expressions can be compiled to expression trees or to delegates, depending on the type that the query is applied to. IEnumerable<T> queries are compiled to delegates. IQueryable and IQueryable<T> queries are compiled to expression trees. For more information, see Expression Trees (C# and Visual Basic).

The following table lists topics that provide additional information about queries and code examples for common tasks.

Topic	Description
Query Expression Basics (C# Programming Guide)	Introduces fundamental query concepts and provides examples of C# query syntax.
How to: Write LINQ Queries in C#	Provides examples of several basic types of query expressions.
How to: Handle Exceptions in Query Expressions (C# Programming Guide)	How and when to move potential exception-throwing code outside a query expression.
How to: Populate Object Collections from Multiple Sources (LINQ)	How to use the **select** statement to merge data from different sources into a new type.
How to: Group Query Results (C# Programming Guide)	Shows different ways to use the **group** clause.
How to: Create a Nested Group (C# Programming Guide)	Shows how to create nested groups.
How to: Perform a Subquery on a Grouping Operation (C# Programming Guide)	Shows how to use a sub-expression in a query as a data source for a new query.
How to: Group Results by Contiguous Keys (C# Programming Guide)	Shows how to implement a thread-safe standard query operator that can perform grouping operations on streaming data sources.
How to: Dynamically Specify Predicate Filters at Runtime (C# Programming Guide)	Shows how to supply an arbitrary number of values to use in equality comparisons in a **where** clause.
How to: Store the Results of a Query in Memory (C# Programming Guide)	Illustrates how to materialize and store query results without necessarily using a **foreach** loop.
How to: Return a Query from a Method (C# Programming Guide)	Shows how to return query variables from methods, and how to pass them to methods as input parameters.
How to: Perform Custom Join Operations (C# Programming Guide)	Shows how to perform join operations based on any kind of predicate function.
How to: Join by Using Composite Keys (C# Programming Guide)	Shows how to join two sources based on more than one matching key.
How to: Order the Results of a Join Clause (C# Programming Guide)	Shows how to order a sequence that is produced by a join operation.
How to: Perform Inner Joins (C# Programming Guide)	Shows how to perform an inner join in LINQ.
How to: Perform Grouped Joins (C# Programming Guide)	Shows how to produce a grouped join in LINQ.
How to: Perform Left Outer Joins (C# Programming Guide)	Shows how to produce a left outer join in LINQ.
How to: Handle Null Values in Query Expressions (C# Programming Guide)	Shows how to handle null values in LINQ queries.

See Also

C# Programming Guide
LINQ (Language-Integrated Query)
Walkthrough: Writing Queries in C# (LINQ)
Basic LINQ Query Operations (C#)
Query Syntax and Method Syntax in LINQ (C#)
Standard Query Operators Overview
Query Keywords (C# Reference)
How Linq to Objects Queries Work
Reading and Writing Queries
What is a collection?
Link to Everything: A List of LINQ Providers