The **WHERE**standard query operator belong to Restriction Operators category in LINQ. Just like SQL, the WHERE standard query operator in LINQ is used to filter rows. The filter expression is specified using a predicate.   
  
   
  
The following are the **2 overloaded versions of WHERE** extension method in Enumerable class

public static IEnumerable<TSource> Where<TSource>(

    this IEnumerable<TSource> source,

    Func<TSource, bool> predicate);

public static IEnumerable<TSource> Where<TSource>(

    this IEnumerable<TSource> source,

    Func<TSource, int, bool> predicate);

**What is a Predicate?**  
A predicate is a function to test each element for a condition  
  
In the following example, the Lambda expression (num => num % 2 == 0) runs for each element in List<int>. If the number is divisible by 2, then a boolean value true is returned otherwise false.

using System;

using System.Collections.Generic;

using System.Linq;

namespace Demo

{

    class Program

    {

        static void Main()

        {

            List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

            IEnumerable<int> evenNumbers = numbers.Where(num => num % 2 == 0);

            foreach (int evenNumber in evenNumbers)

            {

                Console.WriteLine(evenNumber);

            }

        }

    }

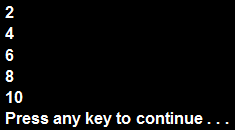
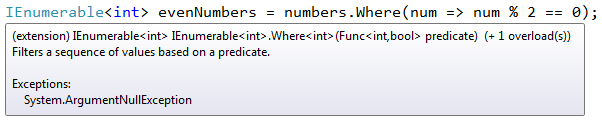
}

// Using SQL like syntax

IEnumerable<int> evenNumbers = from num in numbers

                                                           where num % 2 == 0

                                                           select num;

**Note:**The where query operator is optional.  
  
The program prints all the even numbers   
   
  
When you hover the mouse ove **WHERE**method in the above example, visual studio intellisense shows the following. Notice that in this case, the predicate expects an int input parameter and returns a boolean value. The lambda expression that is passed operates on an int type and should return boolean, otherwise there will be compile time error.   
  
  
**So this means, the line below from the above example**

IEnumerable<int> evenNumbers = numbers.Where(num => num % 2 == 0);

**can be rewritten as shown below**

Func<int, bool> predicate = i => i % 2 == 0;

IEnumerable<int> evenNumbers = numbers.Where(predicate);

**or like below**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Demo

{

    class Program

    {

        static void Main()

        {

            List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

            IEnumerable<int> evenNumbers = numbers.Where(num => IsEven(num));

            foreach (int evenNumber in evenNumbers)

            {

                Console.WriteLine(evenNumber);

            }

        }

        public static bool IsEven(int number)

        {

            if (number % 2 == 0)

            {

                return true;

            }

            else

            {

                return false;

            }

        }

    }

}

**Example 2:**  
The int parameter of the predicate function represents the index of the source element

public static IEnumerable<TSource> Where<TSource>(

    this IEnumerable<TSource> source,

    Func<TSource, int, bool> predicate);

**The following program prints the index position of all the even numbers**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Demo

{

    class Program

    {

        static void Main()

        {

            List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

            IEnumerable<int> evenNumberIndexPositions = numbers

                .Select((num, index) => new { Number = num, Index = index })

                .Where(x => x.Number % 2 == 0)

                .Select(x => x.Index);

            foreach (int evenNumber in evenNumberIndexPositions)

            {

                Console.WriteLine(evenNumber);

            }

        }

    }

}

**Example 3:**  
Use the following SQL to create **Departments**and **Employees**tables

Create table Departments

(

     ID int primary key identity,

     Name nvarchar(50),

     Location nvarchar(50)

)

GO

Create table Employees

(

     ID int primary key identity,

     FirstName nvarchar(50),

     LastName nvarchar(50),

     Gender nvarchar(50),

     Salary int,

     DepartmentId int foreign key references Departments(Id)

)

GO

Insert into Departments values ('IT', 'New York')

Insert into Departments values ('HR', 'London')

Insert into Departments values ('Payroll', 'Sydney')

GO

Insert into Employees values ('Mark', 'Hastings', 'Male', 60000, 1)

Insert into Employees values ('Steve', 'Pound', 'Male', 45000, 3)

Insert into Employees values ('Ben', 'Hoskins', 'Male', 70000, 1)

Insert into Employees values ('Philip', 'Hastings', 'Male', 45000, 2)

Insert into Employees values ('Mary', 'Lambeth', 'Female', 30000, 2)

Insert into Employees values ('Valarie', 'Vikings', 'Female', 35000, 3)

Insert into Employees values ('John', 'Stanmore', 'Male', 80000, 1)

GO

Add an **ADO.NET entity data model**based on the above 2 tables.  
  
Write a LINQ query to retrieve **IT and HR department names and all the male employees**with in these 2 departments.

using System;

using System.Collections.Generic;

using System.Linq;

namespace Demo

{

    class Program

    {

        static void Main()

        {

            EmployeeDBContext context = new EmployeeDBContext();

            IEnumerable<Department> departments = context.Departments

                .Where(dept => dept.Name == "IT" || dept.Name == "HR");

            foreach (Department department in departments)

            {

                Console.WriteLine("Department Name = " + department.Name);

                foreach (Employee employee in department

                    .Employees.Where(emp => emp.Gender == "Male"))

                {

                    Console.WriteLine("\tEmployee Name = " + employee.FirstName

                        + " " + employee.LastName);

                }

                Console.WriteLine();

            }

        }

    }

}

**Output:**   
