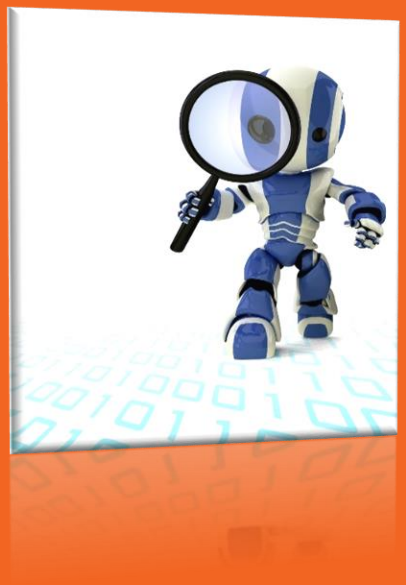
FPT Education

FPT POLYTECHNIC



LẬP TRÌNH C# 2

BÀI 6: LINQ ELEMENT, JOIN, SET

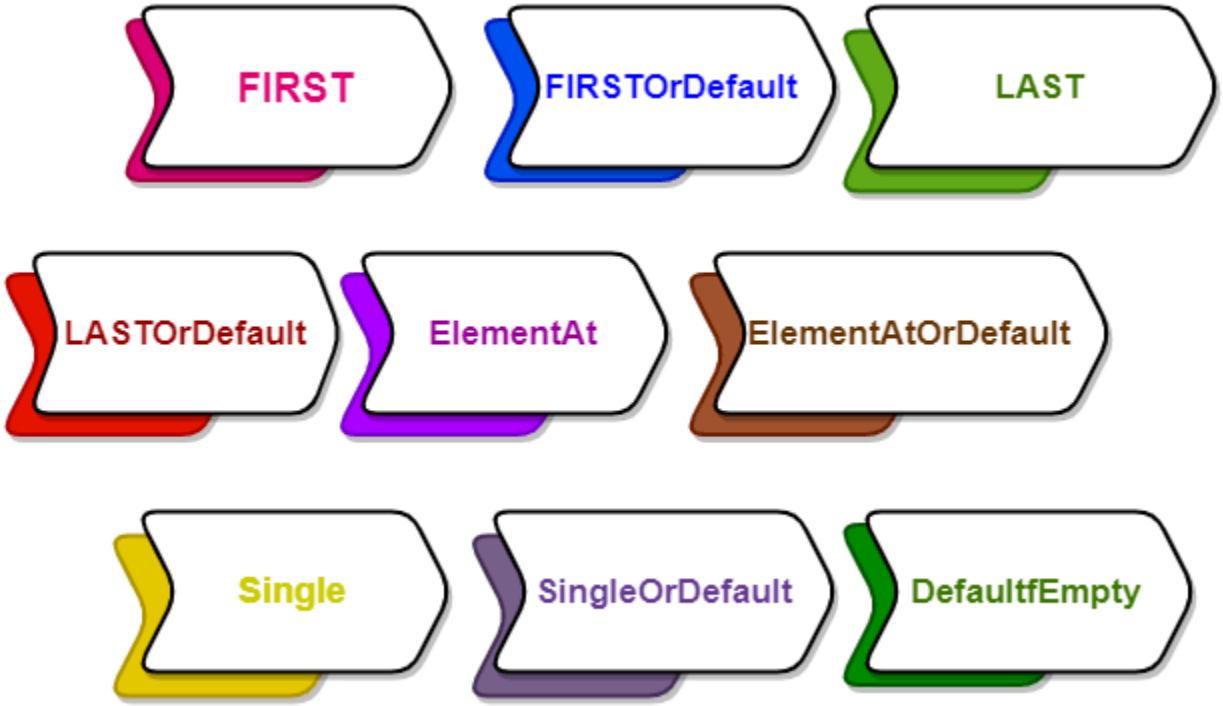
www.poly.edu.vn

Mục tiêu

- LINQ Element Operators
- LINQ Join Operators
- LINQ Set Operations

LINQ Element Operators

- LINQ Element Operators: trả về các phần tử đầu hoặc cuối hoặc theo vị trí được chỉ định trong danh sách/tập hợp



LINQ Element Operators

- LINQ First()
  - Trả về giá trị đầu tiên trong danh sách/tập hợp
  - Trả về Exception nếu không tìm được giá trị thỏa điều kiện hoặc nếu danh sách/tập hợp rỗng

```
static void Main(string[] args)
{
    IList<int> intList = new List<int>() { 7, 10, 21, 30, 45, 50, 87 };
    IList<string> strList = new List<string>() { null, "Two", "Three", "Four", "Five" };
    IList<string> emptyList = new List<string>();
    Console.WriteLine("1st Element in intList: {0}", intList.First());
    Console.WriteLine("1st Even Element in intList: {0}", intList.First(i => i % 2 == 0));
    Console.WriteLine("1st Element in strList: {0}", strList.First());
    Console.WriteLine("emptyList.First() throws an InvalidOperationException");
    Console.WriteLine("------------------------
    Console.WriteLine(emptyList.First());
    Console.ReadLine();
}
```

```
1st Element in intList: 7
1st Even Element in intList: 10
1st Element in strList:
emptyList.First() throws an InvalidOperationException
------------------------------------------------------------

Unhandled Exception: System.InvalidOperationException: Sequence c
   at System.Linq.Enumerable.First[TSource](IEnumerable`1 source)
   at Slide6.Program.Main(String[] args) in A:\E#(Data - NTFS 1)\
Press any key to continue . . .
```

LINQ Element Operators

- LINQ FirstOrDefault
  - Trả về giá trị đầu tiên trong danh sách/tập hợp
  - Trả về giá trị mặc định khi tập hợp/ danh sách rỗng hoặc giá trị không tìm thấy
  - Nếu phần tử đầu tiên trong danh sách/tập hợp là null và sử dụng LINQ FirstOrDefault có điều kiện sẽ gây ra exception

```
static void Main(string[] args)
{
int[] objList = { 1, 2, 3, 4, 5 };
int[] objVals = { };
int result = objList.FirstOrDefault();
int val = objVals.FirstOrDefault();
Console.WriteLine("Element from the List1: {0}", result);
Console.WriteLine("Element from the List2: {0}", val);
Console.ReadLine();
}
}
```

LINQ Element Operators

LINQ FirstOrDefault

```
static void Main(string[] args)
{
int[] objList = { 1, 2, 3, 4, 5 };
int[] objVals = { };
int result = (from l in objList select l).FirstOrDefault();
int val = (from x in objVals
select x).FirstOrDefault();
Console.WriteLine("Element from the List1: {0}", result);
Console.WriteLine("Element from the List2: {0}", val);
Console.ReadLine();
}
}
```

```
List<int> intList = new List<int>() { 7, 10, 21, 30, 45, 50, 87 };
List<string> strList = new List<string>() { null, "Two", "Three", "Four", "Five" }

onsole.WriteLine("1st Element which is greater than 250 in intList: {0}",
                                intList.First( i > 250));

onsole.WriteLine("1st Even Element in intList: {0}",
                                strList.FirstOrDefault(s => s.Contains("T")));
```

LINQ Element Operators

LINQ Last() Method

- Trả về giá trị cuối cùng trong danh sách/tập hợp
- Trả về Exception nếu không tìm được giá trị thỏa điều kiện hoặc nếu danh sách/tập hợp rỗng

```
IList<int> intList = new List<int>() { 7, 10, 21, 30, 45, 50, 87 };
IList<string> strList = new List<string>() { null, "Two", "Three", "Four", "Five" };
IList<string> emptyList = new List<string>();

Console.WriteLine("Last Element in intList: {0}", intList.Last());

Console.WriteLine("Last Even Element in intList: {0}", intList.Last(i => i % 2 == 0));

Console.WriteLine("Last Element in strList: {0}", strList.Last());

Console.WriteLine("emptyList.Last() throws an InvalidOperationException");
Console.WriteLine("------------------------------------------------------------");
Console.WriteLine(emptyList.Last());
```

LINQ Element Operators

- LINQ LastOrDefault
  - Trả về giá trị cuối cùng trong danh sách/tập hợp
  - Trả về giá trị mặc định khi tập hợp/ danh sách rỗng hoặc giá trị không tìm thấy
  - Nếu phần tử cuối cùng trong danh sách/tập hợp là null và sử dụng LINQ LastOrDefault có điều kiện sẽ gây ra exception

LINQ Element Operators

- LINQ LastOrDefault
  - Trả về giá trị cuối cùng trong danh sách/tập hợp
  - Trả về giá trị mặc định khi tập hợp/ danh sách rỗng hoặc giá trị không tìm thấy
  - Nếu phần tử cuối cùng trong danh sách/tập hợp là null và sử dụng LINQ LastOrDefault có điều kiện sẽ gây ra exception

```
static void Main(string[] args)
{
int[] objList = { 1, 2, 3, 4, 5 };
int[] objVals = { };
int result = (from l in objList select l).LastOrDefault();
int val = (from x in objVals select x).LastOrDefault();
Console.WriteLine("Element from the List1: {0}", result);
Console.WriteLine("Element from the List2: {0}", val);
Console.ReadLine();
}
}
```

LINQ Element Operators

LINQ LastOrDefault

```
IList<int> intList = new List<int>() { 7, 10, 21, 30, 45, 50, 87 };
IList<string> strList = new List<string>() { null, "Two", "Three", "Four", "Five" };
IList<string> emptyList = new List<string>();

Console.WriteLine("Last Element in intList: {0}", intList.LastOrDefault());

Console.WriteLine("Last Even Element in intList: {0}",
                                intList.LastOrDefault(i => i % 2 == 0));

Console.WriteLine("Last Element in strList: {0}", strList.LastOrDefault());

Console.WriteLine("Last Element in emptyList: {0}", emptyList.LastOrDefault());
```

```
IList<int> intList = new List<int>() { 7, 10, 21, 30, 45, 50, 87 };
IList<string> strList = new List<string>() { null, "Two", "Three", "Four", "Five" };

Console.WriteLine("Last Element which is greater than 250 in intList: {0}",
                               intList.Last(i => i > 250));

Console.WriteLine("Last Even Element in intList: {0}",
                               strList.LastOrDefault(s => s.Contains("T")));
```

LINQ Element Operators

- LINQ ElementAt(): trả về các phần tử theo chỉ số index trong danh sách/tập hợp
- ElementAtOrDefault: trả về các phần tử theo chỉ số index trong danh sách/tập hợp và trả về giá tri mặc định nếu index không tồn tại

LINQ Element Operators

Ví dụ ElementAt() và ElementAtOrDefault()

```
IList<int> intList = new List<int>() { 10, 21, 30, 45, 50, 87 };
IList<string> strList = new List<string>() { "One", "Two", null, "Four", "Five" };

Console.WriteLine("1st Element in intList: {0}", intList.ElementAt(0));
Console.WriteLine("1st Element in strList: {0}", strList.ElementAt(0));

Console.WriteLine("2nd Element in intList: {0}", intList.ElementAt(1));
Console.WriteLine("2nd Element in strList: {0}", strList.ElementAt(1));

Console.WriteLine("3rd Element in intList: {0}", intList.ElementAtOrDefault(2));
Console.WriteLine("3rd Element in strList: {0}", strList.ElementAtOrDefault(2));

Console.WriteLine("10th Element in intLis
                intList.ElementAtOrDefaul
Console.WriteLine("10th Element in strLis
                 strList.ElementAtOrDefau



Console.WriteLine("intList.ElementAt(9) t
Console.WriteLine("----------------------
Console.WriteLine(intList.ElementAt(9));
```

```
1st Element in intList: 10
1st Element in strList: One
2nd Element in intList: 21
2nd Element in strList:
3rd Element in intList: 30
3rd Element in strList: Three
10th Element in intList: 0 - default int value
10th Element in strList: - default string value (null)
intList.ElementAt(9) throws an exception: Index out of range
------------------------------------------------------------
Run-time exception: Index was out of range....
```

LINQ Element Operators

- LINQ Single():
  - Trả về chỉ một phần tử có hoặc không có điều kiện trong danh sách/tập hợp
  - Trả về exception "InvalidOperationException" nếu không tồn tại phần tử hoặc có nhiều hơn một phần tử thỏa điều kiện

```
static void Main(string[] args)
{
    IList<int> oneElementList = new List<int>() { 7 };
    IList<int> intList = new List<int>() { 8, 17, 21, 30, 45, 50, 87 };
    IList<string> strList = new List<string>() { null, "Two", "Three", "Four", "Five" };
    IList<string> emptyList = new List<string>();
    Console.WriteLine("The only element in oneElementList: {0}", oneElementList.Single());
    //Console.WriteLine("Element in emptyList: {0}", emptyList.SingleOrDefault());
    Console.WriteLine("The only element which is less than 10 in intList: {0}",
                intList.Single(i => i < 10));
}
```

LINQ Element Operators

- LINQ SingleOrDefault():
  - Trả về chỉ một phần tử có hoặc không có điều kiện trong danh sách/tập hợp
  - Trả về exception "InvalidOperationException" nếu có nhiều hơn một phần tử thỏa điều kiện hoặc danh sách/tập hợp chứa nhiều hơn một phần tử
  - Trả về giá trị mặc định khi không có phần tử nào thỏa điều kiện

LINQ Element Operators

- LINQ SingleOrDefault():

```
    IList<int> oneElementList = new List<int>() { 7 };
IList<int> intList = new List<int>() { 7, 10, 21, 30, 45, 50, 87 };
IList<string> strList = new List<string>() { null, "Two", "Three", "Four", "Five" };
IList<string> emptyList = new List<string>();

//following throws error because list contains more than one element which is less than 100
Console.WriteLine("Element less than 100 in intList: {0}", intList.Single(i => i < 100));

//following throws error because list contains more than one element which is less than 100
Console.WriteLine("Element less than 100 in intList: {0}",
                                    intList.SingleOrDefault(i => i < 100));

//following throws error because list contains more than one elements
Console.WriteLine("The only Element in intList: {0}", intList.Single());

//following throws error because list contains more than one elements
Console.WriteLine("The only Element in intList: {0}", intList.SingleOrDefault());

//following throws error because list does not contains any element
Console.WriteLine("The only Element in emptyList: {0}", emptyList.Single());
```

LINQ Element Operators

- LINQ DefaultIfEmpty(): Trả về các giá trị mặc định của một danh sách/tập hợp cho trước rỗng hoặc null. Ngược lại trả về các giá trị của danh sách/tập hợp cho trước.

```
static void Main(string[] args)
{
int[] List1 = { 1,2,3,4,5 };
int[] List2 = { };
var result = List1.DefaultIfEmpty();
var result1 = List2.DefaultIfEmpty();
Console.WriteLine("----List1 with Values----");
foreach (var val1 in result)
{
Console.WriteLine(val1);
}
Console.WriteLine("---List2 without Values---");
foreach (var val2 in result1)
{
Console.WriteLine(val2);
}
Console.ReadLine();
}
}
```

LINQ Element Operators

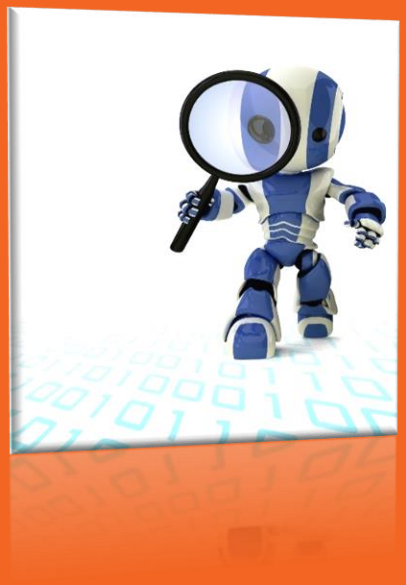
Operator	Description
First	It returns first element in sequence or first element from the collection based on specified condition
FirstOrDefault	Its same as First but it returns default value in case if no element found in collection
Last	It returns last elements in sequence or last element from the list based on matching criteria
LastOrDefault	Its same as Last but it returns default value in case if no element found in collection
ElementAt	It returns an element from the list based on specified index position
ElementAtOrDefault	Its same as ElementAt but it returns default value in case if no element present at specified index of collection
Single	It returns single specific element from the collection
SingleOrDefault	Its same as Single but it returns default value in case if no element found in collection
DefaultIfEmpty	It returns default value in case if the list or collection contains empty or null values

DEMO

- Hiện thực các ví dụ

FPT Education

FPT POLYTECHNIC



LẬP TRÌNH C# 2

BÀI 6: LINQ ELEMENT, JOIN, SET(P2)

www.poly.edu.vn

JOIN OPERATORS

- Sử dụng LINQ Join() Operators kết hợp hai hay nhiều danh sách/tập hợp dựa trên các tiêu chí xác định

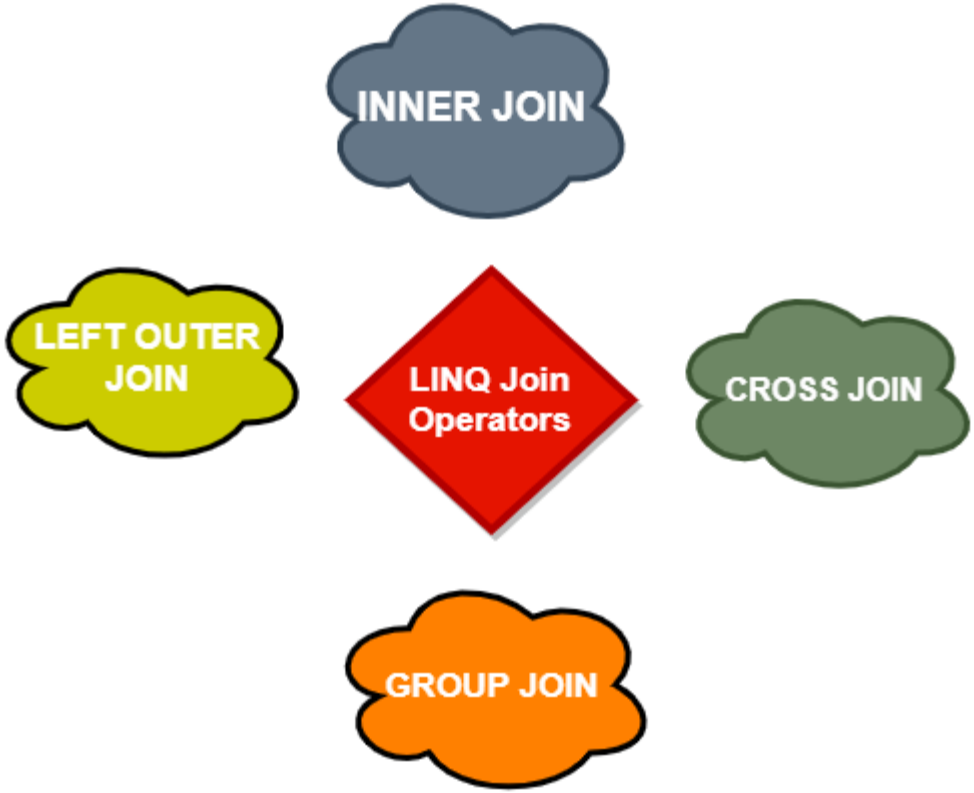
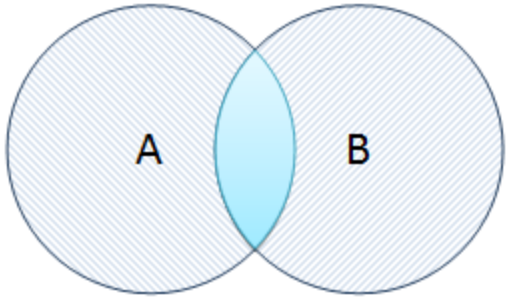


FIG: LINQ JOIN OPERATOR

- LINQ Inner Join
  - Trả về một tập kết quả từ nhiều tập hợp/danh sách
  - Tập kết quả này là giao của các tập hợp được kết dựa trên giá trị (điều kiện) so sánh



JOIN OPERATORS

- LINQ Inner Join

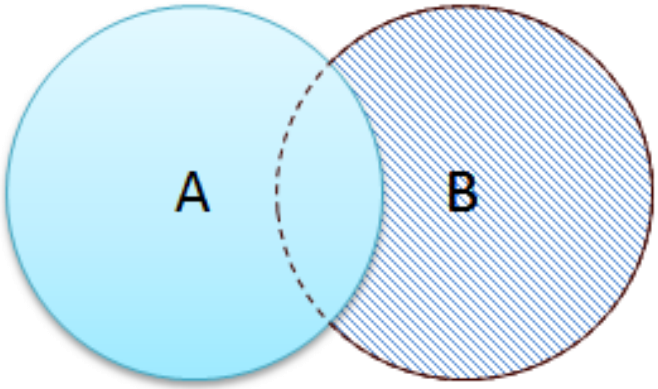
```
class Department
{
public int DepId { get; set; }
public string DepName { get; set; }
}
class Employee
{
public int EmpId { get; set; }
public string Name { get; set; }
public int DeptId { get; set; }
}
```

```
Suresh Dasari      | Software
Rohini Alavala     | Software
Praveen Alavala  | Finance
Sateesh Alavala  | Finance
```

```
static void Main(string[] args)
{
List<Department> objDept = new List<Department>(){
new Department{DepId=1,DepName="Software"},
new Department{DepId=2,DepName="Finance"},
new Department{DepId=3,DepName="Health"}
};
List<Employee> objEmp = new List<Employee>()
{
new Employee { EmpId=1,Name = "Suresh Dasari", DeptId=1 },
new Employee { EmpId=2,Name = "Rohini Alavala", DeptId=1 },
new Employee { EmpId=3,Name = "Praveen Alavala", DeptId=2 },
new Employee { EmpId=4,Name = "Sateesh Alavala", DeptId =2},
new Employee { EmpId=5,Name = "Madhav Sai"}
};
var result = from d in objDept
join e in objEmp
on d.DepId equals e.DeptId
select new
{
EmployeeName = e.Name,
DepartmentName = d.DepName
};
foreach (var item in result)
{
Console.WriteLine(item.EmployeeName + "\t | " + item.DepartmentName);
}
Console.ReadLine();
}
}
```

JOIN OPERATORS

- LINQ Left Join: trả về toàn bộ các dữ liệu trong danh sách bên trái và các dữ liệu bên phải khớp với điều kiện kết



JOIN OPERATORS

- LINQ Left Join:

```
Suresh Dasari     | Software
Rohini Alavala    | Software
Praveen Alavala   | Finance
Sateesh Alavala   | Finance
Madhav Sai        | No Department
```

```
static void Main(string[] args)
{
List<Department> objDept = new List<Department>(){
new Department{DepId=1,DepName="Software"},
new Department{DepId=2,DepName="Finance"},
new Department{DepId=3,DepName="Health"}
}
List<Employee> objEmp = new List<Employee>()
{
new Employee { EmpId=1,Name = "Suresh Dasari", DeptId=1 },
new Employee { EmpId=2,Name = "Rohini Alavala", DeptId=1 },
new Employee { EmpId=3,Name = "Praveen Alavala", DeptId=2 },
new Employee { EmpId=4,Name = "Sateesh Alavala", DeptId =2},
new Employee { EmpId=5,Name = "Madhav Sai"}
}
var result = from e in objEmp
join d in objDept
on e.DeptId equals d.DepId into empDept
from ed in empDept.DefaultIfEmpty()
select new
{
EmployeeName = e.Name,
DepartmentName = ed == null ? "No Department" : ed.DepName
}
foreach (var item in result)
{
Console.WriteLine(item.EmployeeName + "\t | " + item.DepartmentName);
}
Console.ReadLine();
}
}
```

JOIN OPERATORS

- LINQ Group Join
  - Kết hợp join và into, ta có thể kết nhiều tập dữ liệu và kết quả trả về sẽ được nằm trong một cấu trúc phân cấp.
  - Tập dữ liệu bên trái sẽ được lấy tất cả, không quan tâm chúng có được so khớp trùng với tập dữ liệu bên phải hay không

JOIN OPERATORS

- LINQ Group Join

```
Software

    Rohini Alavala
    Suresh Dasari

Finance

    Praveen Kumar
    Sateesh Chandra

Health
```

```
List<Department> objDept = new List<Department>(){
new Department{DepId=1,DepName="Software"},
new Department{DepId=2,DepName="Finance"},
new Department{DepId=3,DepName="Health"}
};
List<Employee> objEmp = new List<Employee>()
{
new Employee { EmpId=1,Name = "Suresh Dasari", DeptId=1 },
new Employee { EmpId=2,Name = "Rohini Alavala", DeptId=1 },
new Employee { EmpId=3,Name = "Praveen Kumar", DeptId=2 },
new Employee { EmpId=4,Name = "Sateesh Chandra", DeptId =2},
new Employee { EmpId=5,Name = "Madhav Sai"}
};
var result = from d in objDept
join e in objEmp on d.DepId equals e.DeptId into empDept
select new
{
DepartmentName = d.DepName,
Employees = from emp2 in empDept
orderby emp2.Name
select emp2
};
int totalItems = 0;
foreach (var empGroup in result)
{
Console.WriteLine(empGroup.DepartmentName);
foreach (var item in empGroup.Employees)
{
totalItems++;
Console.WriteLine("    {0}", item.Name);
}
```

LINQ Set Operations

- Tạo ra một tập kết quả dựa trên sự hiện diện hay vắng mặt của các yếu tố tương đương trong cùng một danh sách hoặc khác danh sách

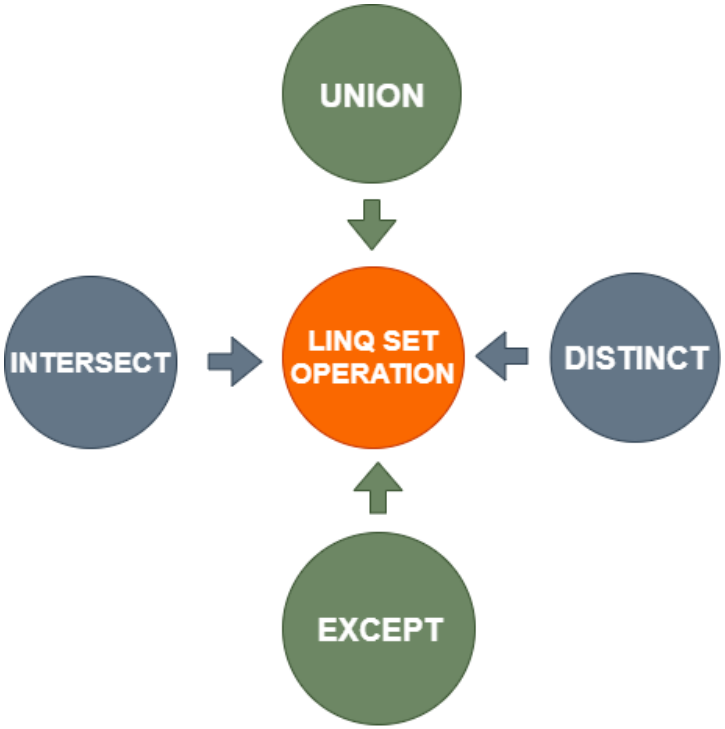
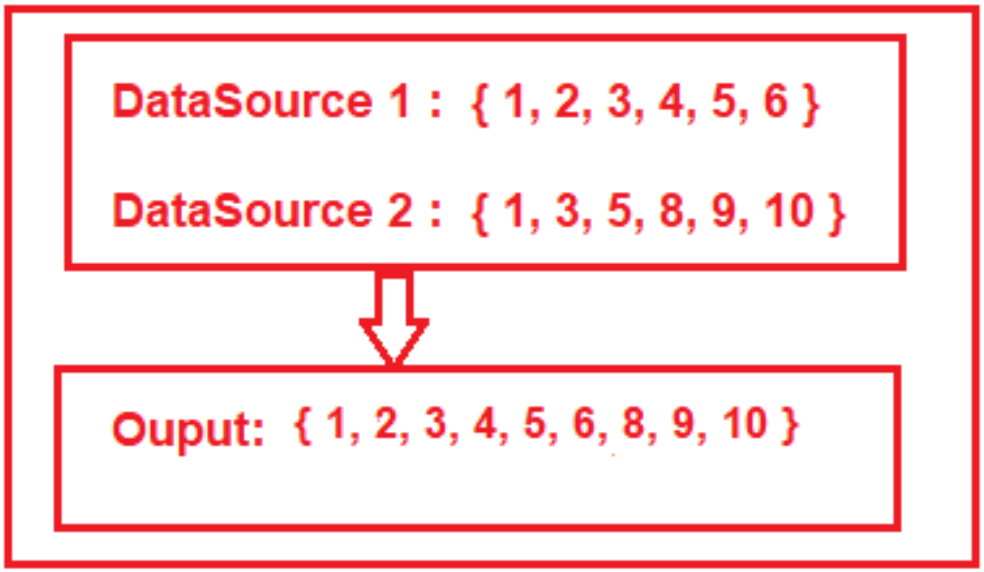


FIG: LINQ SET OPERATIONS

LINQ Set Operations

- LINQ union: trả về tập kết quả kết hợp từ nhiều danh sách, các phần tử chỉ xuất hiện một lần duy nhất trong tập kết quả



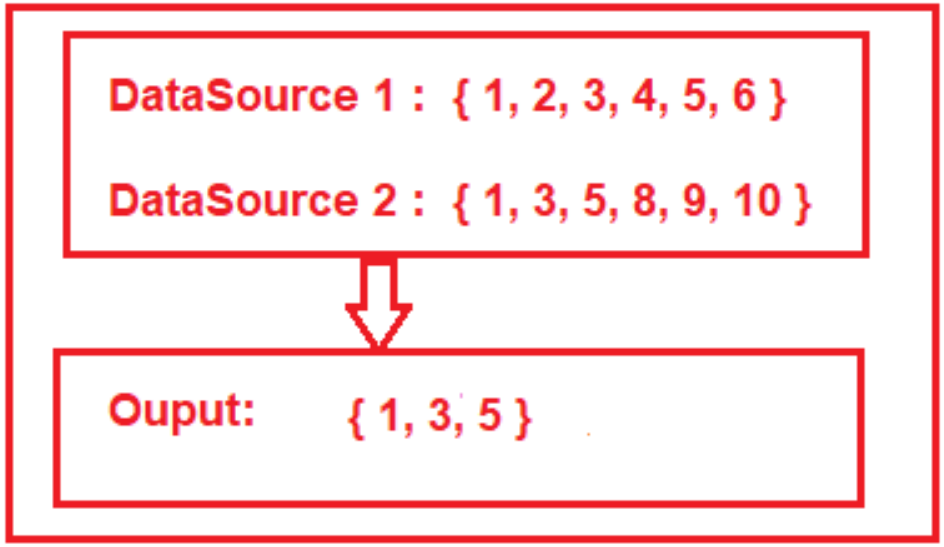
LINQ Set Operations

- LINQ union:

```
static void Main(string[] args)
{
    List<int> dataSource1 = new List<int>() { 1, 2, 3, 4, 5, 6 };
    List<int> dataSource2 = new List<int>() { 1, 3, 5, 8, 9, 10 };

    //Method Syntax
    var MS = dataSource1.Union(dataSource2).ToList();

    //Query Syntax
    var QS = (from num in dataSource1
              select num)
              .Union(dataSource2).ToList();

    foreach (var item in MS)
    {
        Console.WriteLine(item);
    }

    Console.ReadKey();
}
```

LINQ Set Operations

- LINQ Intersect:
  - Trả về kết quả là tập các phần tử đồng thời xuất hiện ở tất cả các danh sách



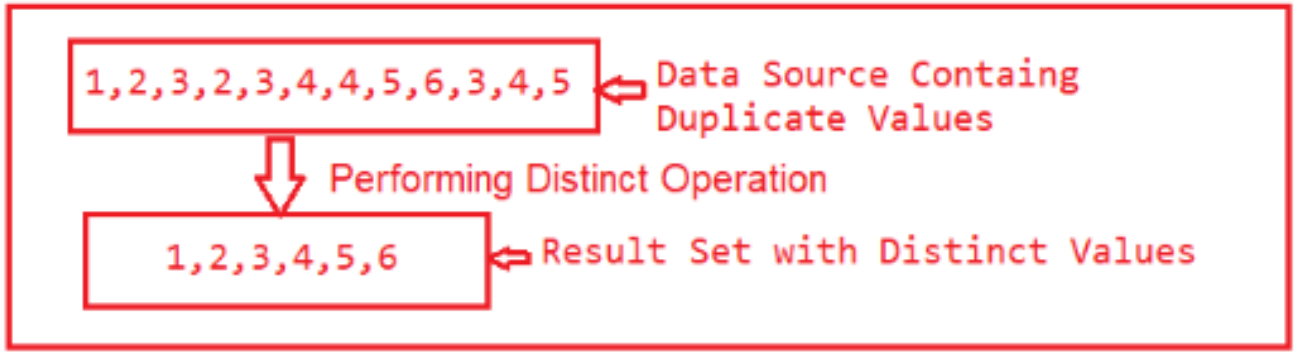
LINQ Set Operations

- LINQ Intersect:
  - Trả về kết quả là tập các phần tử đồng thời xuất hiện ở tất cả các danh sách

```
static void Main(string[] args)
{
    List<int> dataSource1 = new List<int>() { 1, 2, 3, 4, 5, 6 };
    List<int> dataSource2 = new List<int>() { 1, 3, 5, 8, 9, 10 };

    //Method Syntax
    var MS = dataSource1.Intersect(dataSource2).ToList();

    //Query Syntax
    var QS = (from num in dataSource1
              select num)
              .Intersect(dataSource2).ToList();

    foreach (var item in MS)
    {
        Console.WriteLine(item);
    }

    Console.ReadKey();
}
```

LINQ Set Operations

- LINQ Distinct :
  - Trả về kết quả là tập các phần tử không trùng lặp lại




```
List<int> intCollection = new List<int>()
{
    1,2,3,2,3,4,4,5,6,3,4,5
};

//Using Method Syntax
var MS = intCollection.Distinct();

//Using Query Syntax
var QS = (from num in intCollection
          select num).Distinct();
foreach (var item in MS)
{
    Console.WriteLine(item);
}
```

DEMO

- Hiện thực các ví dụ

Tổng kết bài học

- LINQ Element Operators
- LINQ Join Operators
- LINQ Set Operations

KẾT THÚC