LINQ

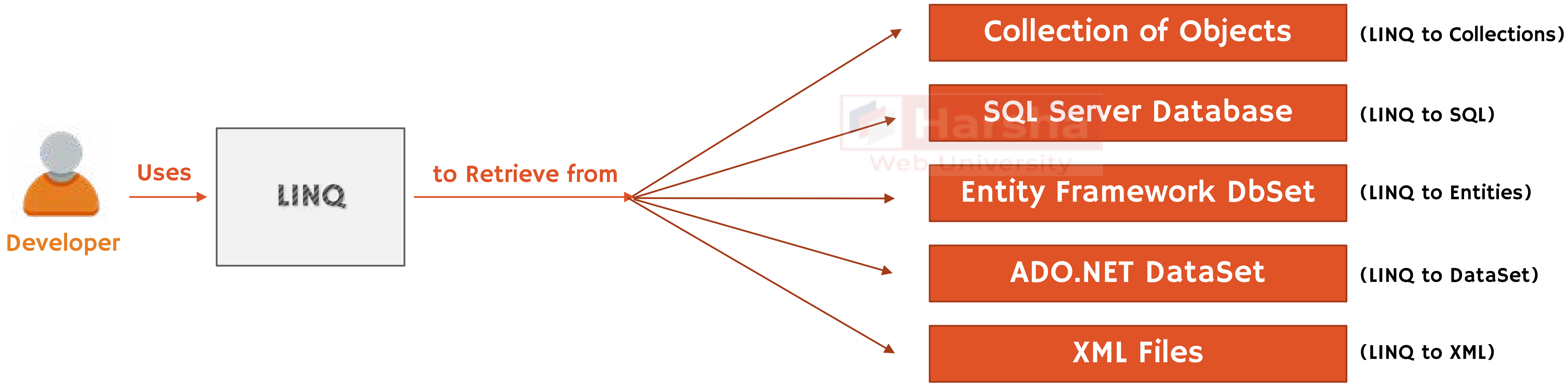
C#

Understanding LINQ

Harsha

What

- LINQ is a 'uniform query syntax' that allows you to retrieve data from various data sources such as arrays, collections, databases, XML files.



Developer

Uses

LINQ

to Retrieve from

Collection of Objects	(LINQ to Collections)
SQL Server Database	(LINQ to SQL)
Entity Framework DbSet	(LINQ to Entities)
ADO.NET DataSet	(LINQ to DataSet)
XML Files	(LINQ to XML)

How

LINQ Query - Example

```
var  result = Customers.Where(temp => temp.Location == "New York").ToList( );
//returns a list of customers from New York location.
```

- **Single Syntax - To Query Multiple Data Sources**
  - Developer uses the same LINQ syntax to retrieve information from various data sources such as collections, SQL Server database, Entity Framework DbSet's, ADO.NET DataSet etc.
- **Compile-Time Checking of Query Errors**
  - Errors in the LINQ query will be identified while compilation time / while writing the code in Visual Studio.
- **IntelliSence Support**
  - The list of properties of types are shown in VS IntelliSence while writing the LINQ queries.

LINQ Extension Methods

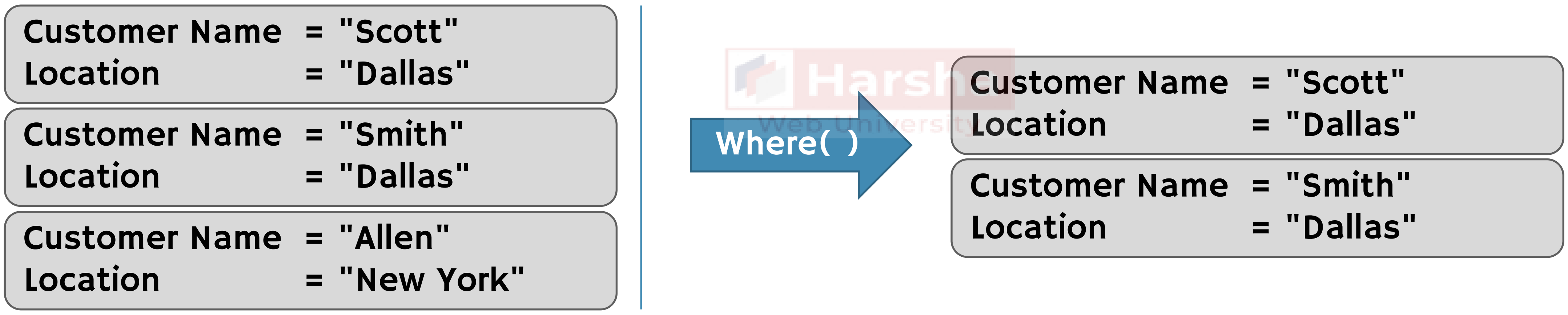
Classification	LINQ Extension Methods / LINQ Operators
Filtering	Where, OfType
Sorting	OrderBy, OrderByDescending, ThenBy, ThenByDescending, Reverse
Grouping	GroupBy
Join	Join
Project	Select, SelectMany
Aggregation	Average, Count, Max, Min, Sum
Quantifiers	All, Any, Contains
Elements	ElementAt, ElementAtOrDefault, First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault
Set Operations	Distinct, Except, Intersect, Union
Partitioning	Skip, SkipWhile, Take, TakeWhile
Concatenation	Concat
Equality	SequenceEqual
Generation	DefaultEmpty, Empty, Range, Repeat
Conversion	AsEnumerable, AsQueryable, Cast, ToArray, ToDictionary, ToList

Where

What

- Where() method filters collection based on given lambda expression and returns a new collection with matching element.

Example



How

Where Extension Method - Declaration

```
Where(Func<TSource, bool>  predicate)
```

Where Extension Method - Usage

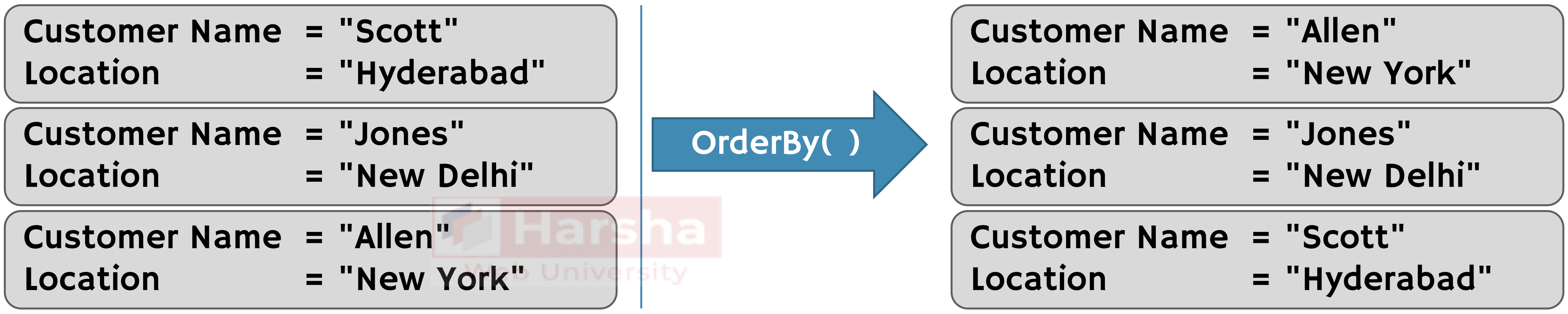
```
var  result = Customers.Where(temp => temp.Location == "Dallas").ToList( );
//returns a list of customers from Hyderabad location.
```

OrderBy

What

- OrderBy() method sorts collection based on given lambda expression (property) and returns a new collection with sorted elements.

Example



Customer Name = "Scott"
Location = "Hyderabad"

Customer Name = "Jones"
Location = "New Delhi"

Customer Name = "Allen"
Location = "New York"

OrderBy( )

Customer Name = "Allen"
Location = "New York"

Customer Name = "Jones"
Location = "New Delhi"

Customer Name = "Scott"
Location = "Hyderabad"

How

OrderBy Extension Method - Declaration

```
OrderBy(Func<TSource, TKey>  keySelector)
```

OrderBy Extension Method - Usage

```
var  result = Customers.OrderBy(temp => temp.CustomerName).ToList( );
//returns a list of customers sorted based on customer name.
```

OrderBy Descending

OrderByDescending Extension Method - Declaration

```
OrderByDescending(Func<TSource, TKey>  keySelector)
```

OrderByDescending Extension Method - Usage

```
var  result = Customers.OrderByDescending(temp => temp.CustomerName).ToList( );
//returns a list of customers sorted based on customer name in descending order.
```

ThenBy

ThenBy Extension Method - Declaration

```
ThenBy(Func<TSource, TKey>  keySelector)
```

ThenBy Extension Method - Usage

```
var  result = Customers.OrderBy(temp => temp.Location)
                      .ThenBy(temp => temp.CustomerName).ToList( );
//returns a list of customers sorted based on location and customer name.
```

TheBy
Descending

ThenByDescending Extension Method - Declaration

```
ThenByDescending(Func<TSource, TKey>  keySelector)
```

ThenByDescending Extension Method - Usage

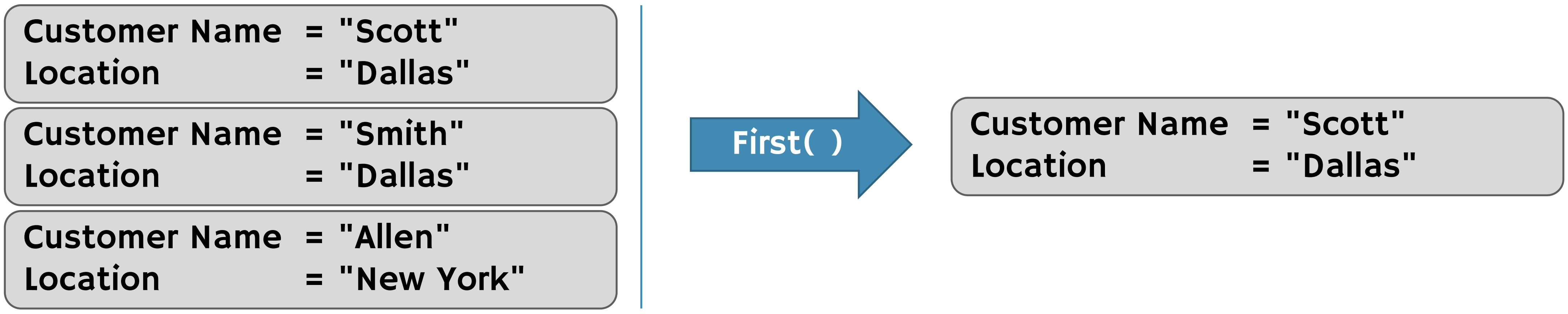
```
var  result = Customers.OrderBy(temp => temp.Location)
                       .ThenByDescending(temp => temp.CustomerName).ToList( );
//returns a list of customers sorted based on location (ascending) and customer name (descending).
```

First

What

- First() method returns first element in the collection that matches with the collection.
- It throws exception if no element matches with the condition.

Example



Customer Name = "Scott"
Location = "Dallas"

Customer Name = "Smith"
Location = "Dallas"

Customer Name = "Allen"
Location = "New York"

Customer Name = "Scott"
Location = "Dallas"

How

First Extension Method - Declaration

```
First(Func<TSource, bool>  predicate)
```

First Extension Method - Usage

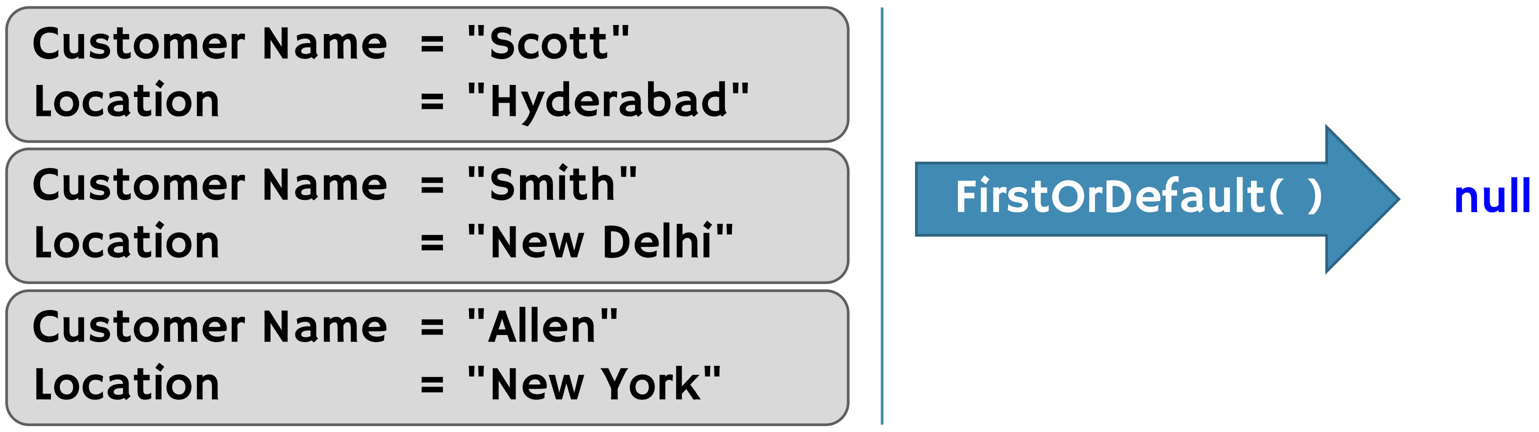
```
var  result = Customers.First(temp => temp.Location == "Dallas");
//returns the first customer from Dallas location.
```

FirstOrDefault

What

- FirstOrDefault() method returns first element that matches with the condition.
- It returns null if no element matches with the condition.

Example



Customer Name = "Scott"
Location = "Hyderabad"

Customer Name = "Smith"
Location = "New Delhi"

Customer Name = "Allen"
Location = "New York"

How

FirstOrDefault Extension Method - Declaration

```
FirstOrDefault(Func<TSource, bool>  predicate)
```

FirstOrDefault Extension Method - Usage

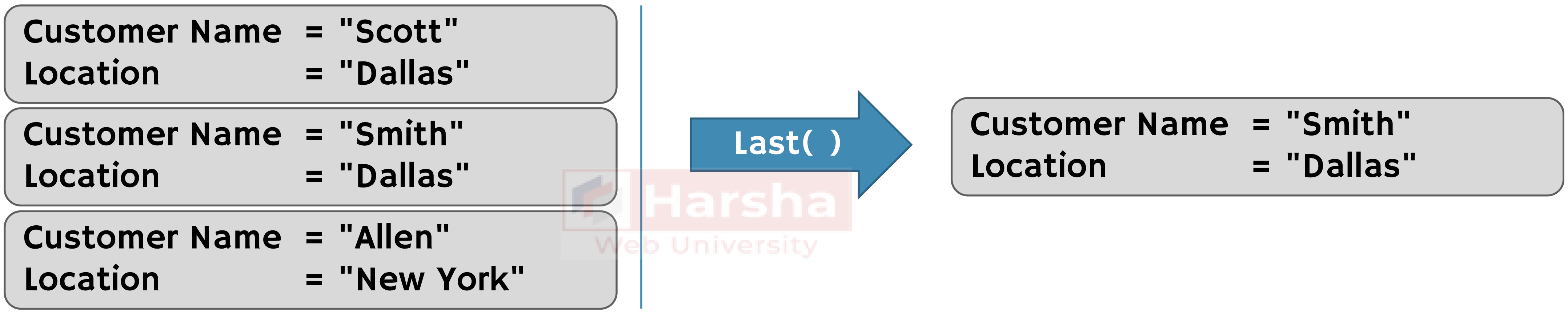
```
var  result = Customers.FirstOrDefault(temp => temp.Location == "London");
//returns the first customer from London location (or) returns null if not exists.
```

Last

What

- Last() method returns last element in the collection that matches with the collection.
- It throws exception if no element matches with the condition.

Example



Customer Name = "Scott"
Location = "Dallas"

Customer Name = "Smith"
Location = "Dallas"

Customer Name = "Allen"
Location = "New York"

Customer Name = "Smith"
Location = "Dallas"

How

Last Extension Method - Declaration

```
Last(Func<TSource, bool>  predicate)
```

Last Extension Method - Usage

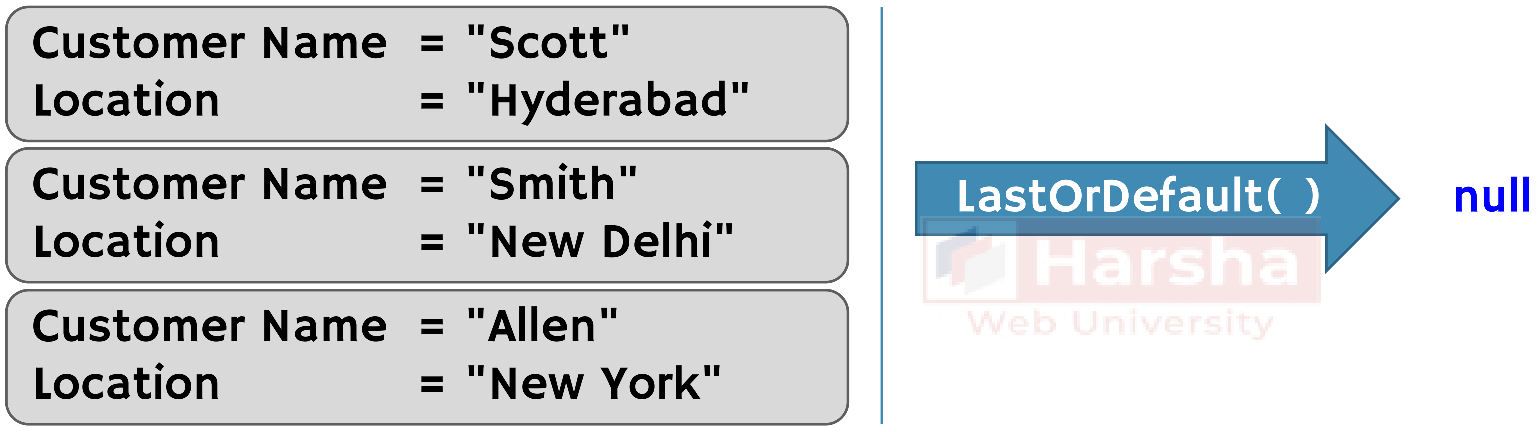
```
var  result = Customers.Last(temp => temp.Location == "Dallas");
//returns the last customer from Dallas location.
```

LastOrDefault

What

- LastOrDefault() method returns last element that matches with the condition.
- It returns null if no element matches with the condition.

Example



Customer Name = "Scott"
Location = "Hyderabad"

Customer Name = "Smith"
Location = "New Delhi"

Customer Name = "Allen"
Location = "New York"

How

LastOrDefault Extension Method - Declaration

```
LastOrDefault(Func<TSource, bool>  predicate)
```

LastOrDefault Extension Method - Usage

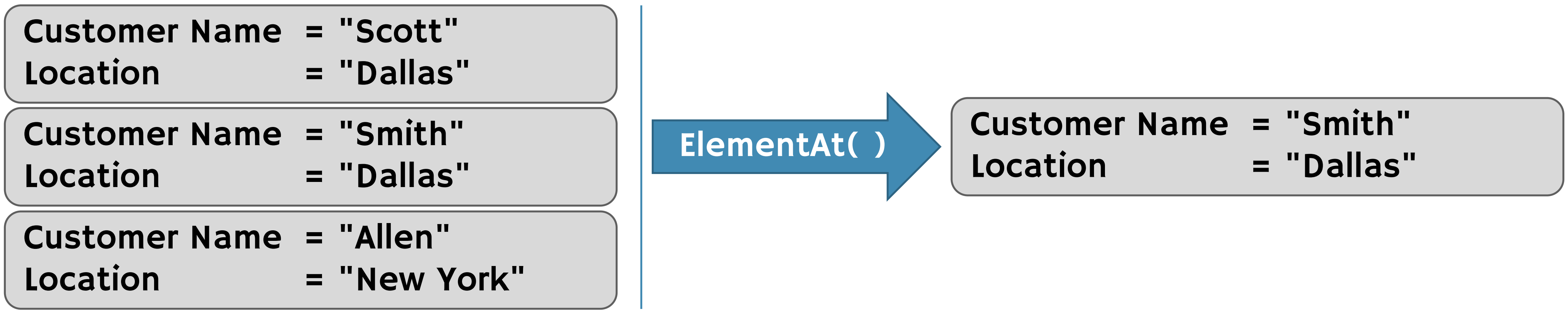
```
var  result = Customers.LastOrDefault(temp => temp.Location == "London");
//returns the last customer from London location (or) returns null if not exists.
```

ElementAt

What

- Element() method returns an element in the collection at specified index.
- It throws exception if no element exists at the specified index; to get 'null' instead, use ElementOrDefault().

Example



Customer Name = "Scott"
Location = "Dallas"

Customer Name = "Smith"
Location = "Dallas"

Customer Name = "Allen"
Location = "New York"

Customer Name = "Smith"
Location = "Dallas"

How

ElementAt Extension Method - Declaration

```
ElementAt(int index)
```

ElementAt Extension Method - Usage

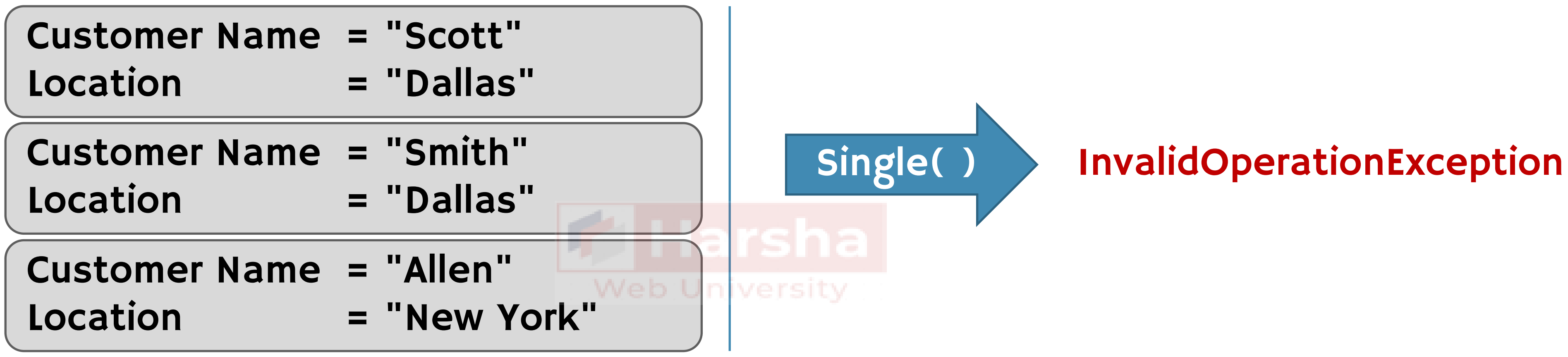
```
var  result = Customers.ElementAt(1);   //returns the customer at index 1
```

Single

What

- It returns first element (only one element) that matches with the collection.
- It throws exception if no element or multiple elements match with the condition.

Example



Customer Name = "Scott"
Location = "Dallas"

Customer Name = "Smith"
Location = "Dallas"

Customer Name = "Allen"
Location = "New York"

Single( ) → InvalidOperationException

How

Single Extension Method - Declaration

```
Single(Func<TSource, bool>  predicate)
```

Single Extension Method - Usage

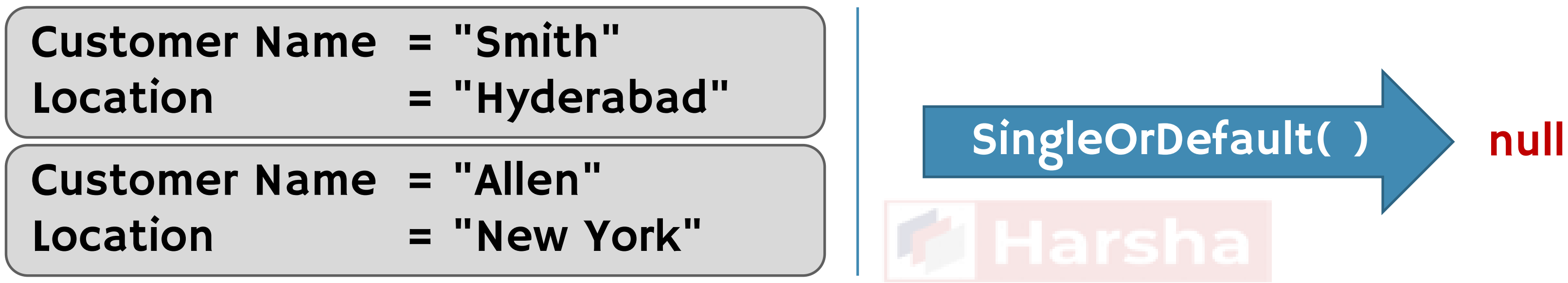
```
var  result = Customers.Single(temp => temp.Location == "Dallas");
//returns the first (only one customer) from Dallas location.
but it throws exception if none / multiple elements matches with the condition.
```

SingleOrDefault

What

- It returns first element (only one element) that matches with the collection.
- It returns null if no element matches with the condition; but it throws exception if multiple elements match with the condition.

Example



Customer Name = "Smith"
Location = "Hyderabad"

Customer Name = "Allen"
Location = "New York"

SingleOrDefault( ) → null

How

SingleOrDefault Extension Method - Declaration

```
SingleOrDefault(Func<TSource, bool>  predicate)
```

SingleOrDefault Extension Method - Usage

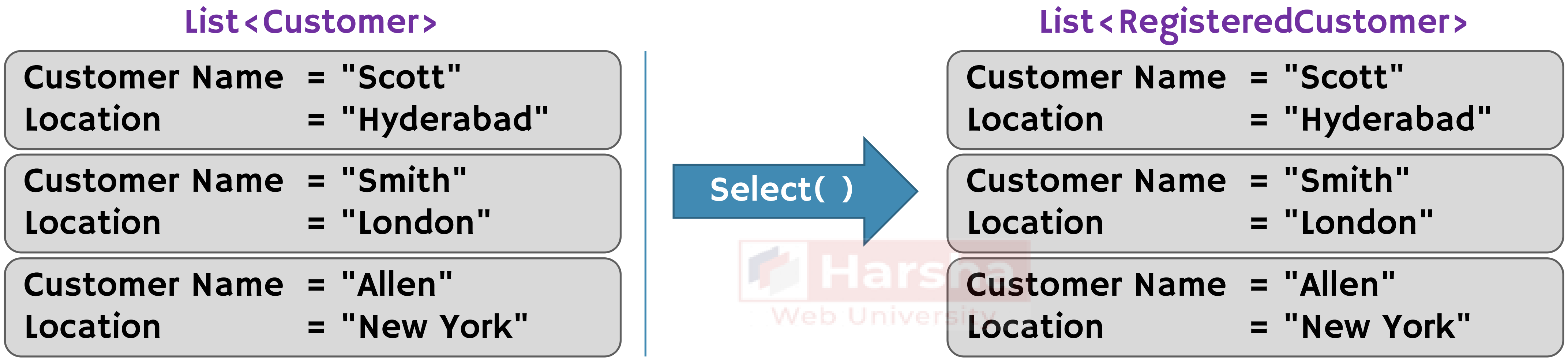
```
var  result = Customers.SingleOrDefault(temp => temp.Location == "London");
//returns the first (only one customer) from London location.
it throws exception if multiple elements matches with the condition; but null in case of no match.
```

Select

What

- It returns collection by converting each element into another type, based on the conversion expression.

Example



List<Customer>

Customer Name = "Scott"
Location = "Hyderabad"

Customer Name = "Smith"
Location = "London"

Customer Name = "Allen"
Location = "New York"

Select( )

List<RegisteredCustomer>

Customer Name = "Scott"
Location = "Hyderabad"

Customer Name = "Smith"
Location = "London"

Customer Name = "Allen"
Location = "New York"

How

Select Extension Method - Declaration

```
Select(Func<TSource, TResult>  selector)
```

Select Extension Method - Usage

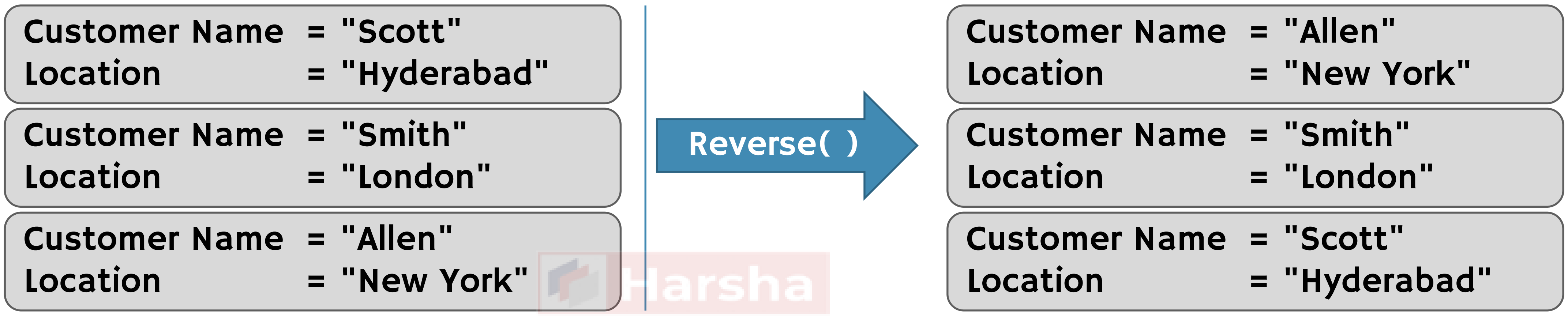
```
var  result = Customers.Select(temp => new RegisteredCustomer()
        { CustomerName = temp.CustomerName, Location = temp.Location } );
//converts all customers into a collection of RegisteredCustomer class.
```

Reverse

What

- It reverses the collection.

Example



Customer Name = "Scott"
Location = "Hyderabad"

Customer Name = "Smith"
Location = "London"

Customer Name = "Allen"
Location = "New York"

Reverse( )

Customer Name = "Allen"
Location = "New York"

Customer Name = "Smith"
Location = "London"

Customer Name = "Scott"
Location = "Hyderabad"

How

Reverse Extension Method - Declaration

```
Reverse( )
```

Reverse Extension Method - Usage

```
var  result = Customers.Reverse( );     //reverses the customers collection
```

Min, Max, Count, Sum, Average

What

- It performs aggregate operations such as finding minimum value of specific property of all elements of a collection.

How

Min, Max, Count, Sum, Average - Example

```
var  result1 = Students.Min( temp => temp.Marks );     //minimum value of Marks property
var  result2 = Students.Max( temp => temp.Marks );    //maximum value of Marks property
var  result3 = Students.Count( );                      //count of elements
var  result4 = Students.Sum( temp => temp.Marks);     //sum value of Marks property
var  result5 = Students.Average( temp => temp.Marks);   //average value of Marks property
```