Using LINQ Select and Order Operations

Paul D. Sheriff
Business/IT Consultant PDS Consulting

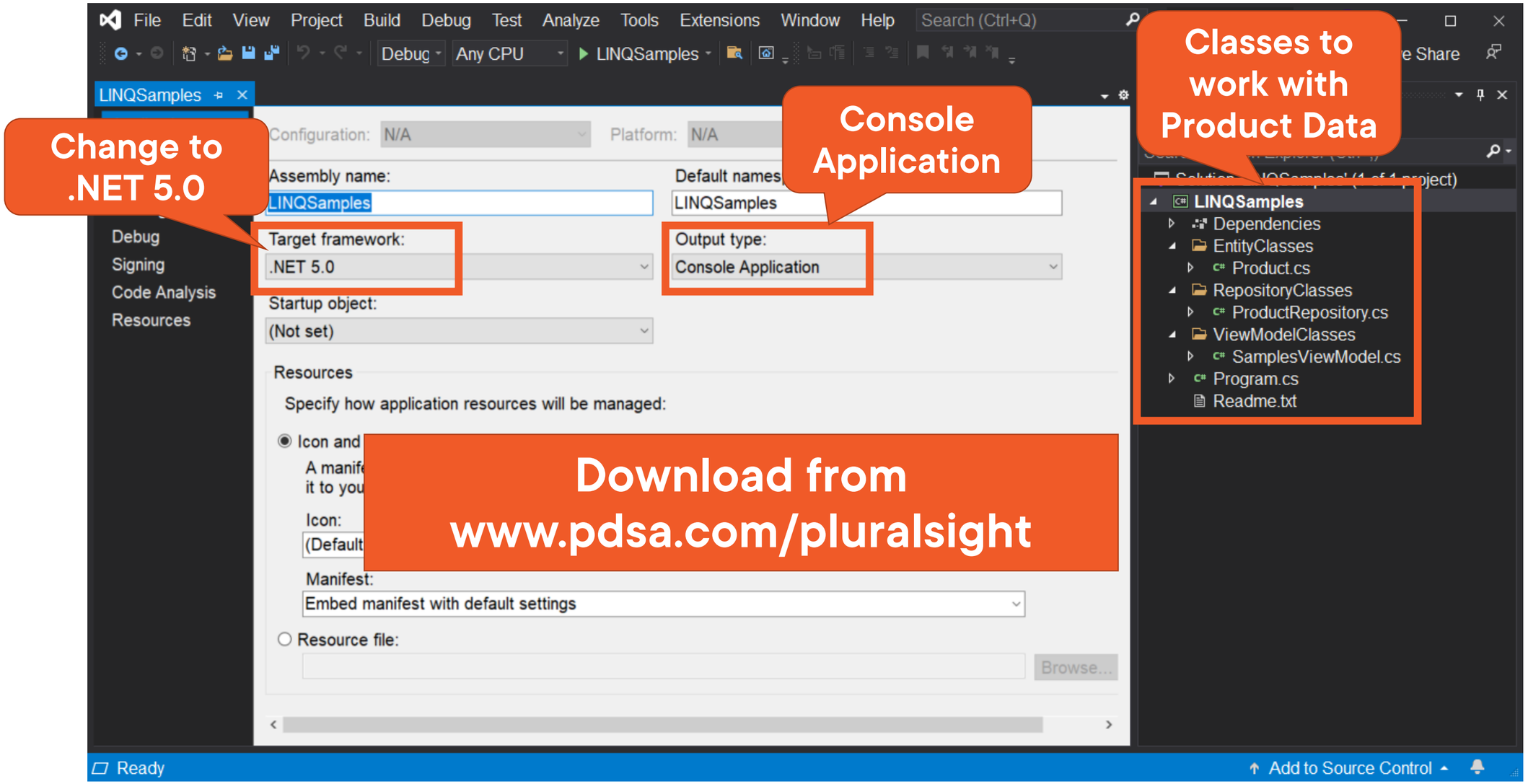
psheriff@pdsa.com www.pdsa.com

Module Goals

- An overview of the console application
- Selecting all data
- Selecting specific columns
- Building an anonymous class
- Ordering data

The Demo Classes

Sample Console Application



Product Entity Class

```
public partial class Product {

  public int ProductID

  public string Name

  public string Color

  public decimal StandardCost

  public decimal ListPrice

  public string Size

}
```

- **Represents a "Product"**
- **Each property would have a { get; set; }**
- **Eliminated here for brevity**

Product Repository Class

```
public partial class ProductRepository {
  public static List<Product> GetAll()  {
    return new List<Product> {
      new Product {
        ProductID = 680,
        Name = "HL Road Frame",
        Color = "Black",
        StandardCost = 1059.31M,
        ListPrice = 1431.50M,
        Size = "58",
      },
  ...
```

- **Class to retrieve collection of product data**
- **Method to retrieve all products from a data source**
- **Just using hard-coded values for this sample**

View Model Classes

```
public class SamplesViewModel {
  public SamplesViewModel()  {
    Products = ProductRepository.GetAll();
  }
  public bool UseQuerySyntax = true;
  public List<Product> Products
  public string ResultText
  public void GetAll() {
  }
  public void GetSingleColumn() {
  }
}
```

- Class used to teach LINQ samples
- Constructor builds a List<Product> objects
- Use query or method syntax?
- Collection of product objects
- Any textual information to display
- Method to retrieve all data from collection
- Method to get a single column of data from collection

View Model Methods

```
public void GetAll() {
  List<Product> list;
  if (UseQuerySyntax) {
    // Query Syntax
    list = (from prod in Products
            select prod).ToList();
  }
  else {
    // Method Syntax
    list = Products.Select(
               prod => prod).ToList();
  }
  ResultText = $"Total Products: {list.Count}";
}
```

- **Method to retrieve all data from collection**
- **Use query syntax?**
- **Write a LINQ query**
- **Use method syntax?**
- **Write a query using LINQ methods**
- **Set informational text**

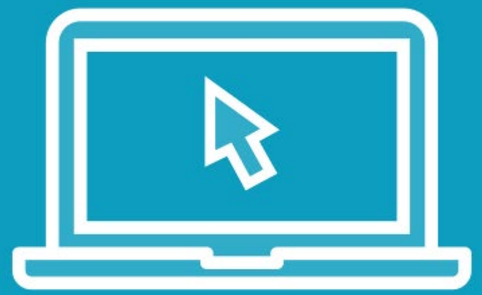
Program.cs

```
static void Main(string[] args) {

  // Instantiate the ViewModel

  SamplesViewModel vm =
    new SamplesViewModel();

  // Call a sample method

  vm.GetAll();

  // Display Product Collection

  foreach (var item in vm.Products) {

    Console.Write(item.ToString());

  }

  // Display Result Text

  Console.WriteLine(vm.ResultText);

}
```

- **Main() runs in the console application**
- **Create instance of view model you wish to use**
- **Call the method to try out**
- **Display all items in the console window**
- **Display any informational results**

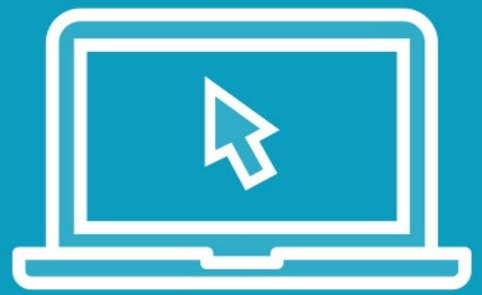
Selecting

Demo



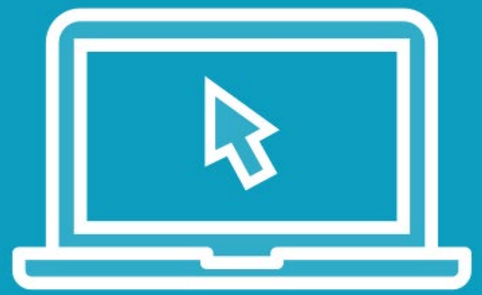
- **Select all items using a loop**

Demo



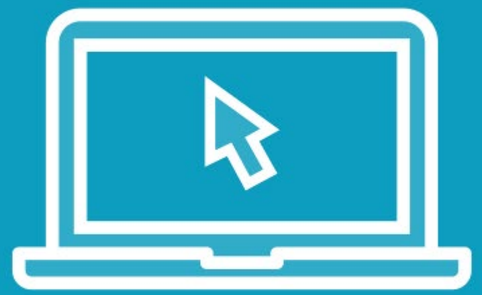
- **Select all items using LINQ**

Demo



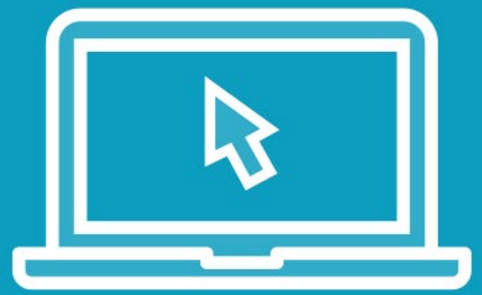
- **Get a single column**

Demo



- **Get specific columns**

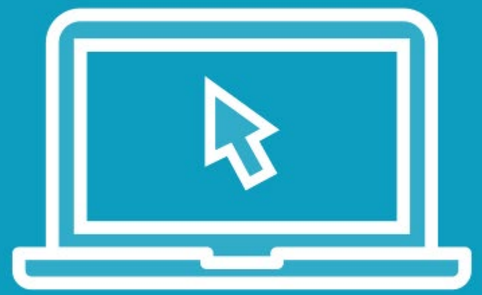
Demo



- **Build an anonymous class**

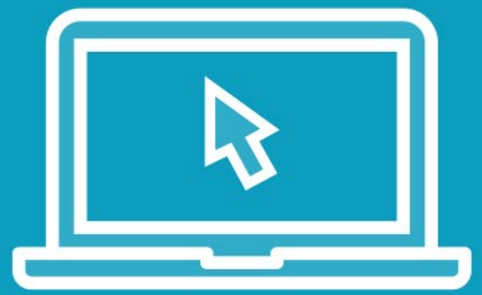
Ordering

Demo



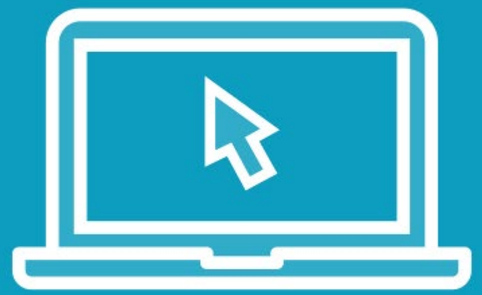
- **Ordering data**

Demo



- **Ordering data in descending order**

Demo



- **Ordering data by two fields**

Module Summary

- Query syntax is very readable
- Method syntax is very precise
- Project new columns using anonymous classes
- Method syntax for ordering is different
  - OrderBy()
  - ThenBy()
  - OrderByDescending()
  - ThenByDescending()

Up Next:

Extract Data Using Filtering and Element Operations