**CRUD and serialization on list collection using class library in c#.net**

**List collection**

List implements the IList interface. List is one of the most flexible data structures of C# collection. Collection classes are special classes for data storage and retrieval. List is of non-generic type which allow you to store any type of data in it.

**Class library**

We can use class library whenever we want to share same data across multiple application which allow to increase reusability of class library. Using class library will give slight load time overhead because externally loaded assemblies are slower than internal loaded assemblies.

**How to create class library**

First create solution

File **>** new **>** project **>** Other Project Types **>** Visual Studio Solution.

Create Class library

Right click on solution **>** add **>** new Project **>** Class Library **>** ClassLibraryProduct(Name).

In program.cs file add following code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Runtime.Serialization;

namespace ClassLibraryProduct

{

[Serializable()]

public class ProductModel

{

private int \_ProductId;

private string \_ProductName;

private string \_Category;

private int \_Price;

private int \_Qty;

private int \_TotalAmount;

public ProductModel(int id,string name,string cat,int price,int qty,int totalamt)

{

\_ProductId = id;

\_ProductName = name;

\_Category = cat;

\_Price = price;

\_Qty = qty;

\_TotalAmount = totalamt;

}

public int ProductId { get { return \_ProductId; } set { \_ProductId = value; } }

public string ProductName { get { return \_ProductName; } set { \_ProductName = value; } }

public string Category { get { return \_Category; } set { \_Category = value; } }

public int Price { get { return \_Price; } set { \_Price = value; } }

public int Qty { get { return \_Qty; } set { \_Qty = value; } }

public int TotalAmount { get { return \_TotalAmount; } set { \_TotalAmount = value; } }

}

}

**Explanation**

In this example we have created ProductModel class and as we want to serialize it, we have used [Serializable()]. For Serializable we have to export using System.Runtime.Serialization namespace.

We have declared 6 private members and to use these private members outside class we have used properties. If you don’t have basic knowledge of properties you can visit following link:

<https://www.c-sharpcorner.com/UploadFile/3d39b4/property-in-C-Sharp/>

Once you have created class library we can use it in console application.

**Create console application**

Right click on solution **>** add **>** new Project **>** Console Application **>** ConsoleApplicationProduct(Name).

After you have created console application, provide reference of class library to console application.

ConsoleApplicationProduct(Name) > Reference > Add Reference > ClassLibraryProduct

Once reference is added, add following namespace in program.cs.

using ClassLibraryProduct;

In program.cs add following code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using ClassLibraryProduct;

using System.Threading;

using System.Runtime.Serialization.Formatters.Binary;

using System.IO;

namespace ConsoleApplicationProduct

{

class Program

{

static void Main(string[] args)

{

int ch=1;

int ID, PRICE, QTY, TOTALAMT;

string NAME, CAT;

List<ProductModel> pro1 = new List<ProductModel>();

while (ch != 0)

{

Console.WriteLine("1.Add product");

Console.WriteLine("2.Display Product");

Console.WriteLine("3.Edit Product");

Console.WriteLine("4.Delete Product");

Console.WriteLine("5.Search Product by Name");

Console.WriteLine("6.Serialize Data");

Console.WriteLine("0.Exit");

Console.WriteLine("Enter your choice");

ch = Convert.ToInt16(Console.ReadLine());

switch (ch)

{

case 1:

Console.WriteLine("Add Product Id");

ID = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Add Product Name");

NAME = Console.ReadLine();

Console.WriteLine("Add Product Category");

CAT = Console.ReadLine();

Console.WriteLine("Add Product price");

PRICE = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Add Product Qty");

QTY = Convert.ToInt16(Console.ReadLine());

TOTALAMT = PRICE \* QTY;

ProductModel p1 = new ProductModel(ID, NAME, CAT, PRICE, QTY,TOTALAMT);

pro1.Add(p1);

break;

case 2:

Console.WriteLine("Id " + " Name " + " Category " + " Price " + " Qty " + " Amount");

foreach (ProductModel pro in pro1)

{

Console.WriteLine(pro.ProductId + " " + pro.ProductName + " " + pro.Category + " " + pro.Price + " " + pro.Qty + " " + pro.TotalAmount);

}

break;

case 3:

int uid;

int UPRICE, UQTY,UTOT;

string UNAME, UCAT;

Console.WriteLine("Enter Product Id to Updte");

uid = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Add Product Name");

UNAME = Console.ReadLine();

Console.WriteLine("Add Product Category");

UCAT = Console.ReadLine();

Console.WriteLine("Add Product price");

UPRICE = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Add Product Qty");

UQTY = Convert.ToInt16(Console.ReadLine());

UTOT = UPRICE \* UQTY;

foreach (ProductModel pro in pro1)

{

if (pro.ProductId == uid)

{

pro.ProductName = UNAME;

pro.Category = UCAT;

pro.Price = UPRICE;

pro.Qty = UQTY;

pro.TotalAmount = UTOT;

}

}

Console.WriteLine("Id " + " Name " + " Category " + " Price " + " Qty " + " Amount");

foreach (ProductModel pro in pro1)

{

Console.WriteLine(pro.ProductId + " " + pro.ProductName + " " + pro.Category + " " + pro.Price + " " + pro.Qty + " " + pro.TotalAmount);

}

break;

case 4:

int did;

Console.WriteLine("Enter Product Id to Delete");

did = Convert.ToInt16(Console.ReadLine());

foreach (ProductModel pro in pro1)

{

if (pro.ProductId == did)

{

pro1.Remove(pro);

break;

}

}

Console.WriteLine("Id " + " Name " + " Category " + " Price " + " Qty " + " Amount");

foreach (ProductModel pro in pro1)

{

Console.WriteLine(pro.ProductId + " " + pro.ProductName + " " + pro.Category + " " + pro.Price + " " + pro.Qty + " " + pro.TotalAmount);

}

break;

case 5:

int sid;

Console.WriteLine("Enter Product Id to Search");

sid = Convert.ToInt16(Console.ReadLine());

foreach (ProductModel pro in pro1)

{

if (pro.ProductId == sid)

{

Console.WriteLine("ProId=" + pro.ProductId);

Console.WriteLine("ProName=" + pro.ProductName);

Console.WriteLine("ProCategory=" + pro.Category);

Console.WriteLine("ProPrice=" + pro.Price);

Console.WriteLine("ProQty" + pro.Qty);

Console.WriteLine("ProAmount" + pro.TotalAmount);

}

}

break;

case 6:

BinaryFormatter bf = new BinaryFormatter();

FileStream fsout = new FileStream("ProductBinary.binary", FileMode.Create, FileAccess.Write);

try

{

using (fsout)

{

bf.Serialize(fsout, pro1);

Console.WriteLine("Serialize");

}

}

catch

{

Console.WriteLine("Not Serialize");

}

break;

case 0:

break;

}

}

}

}

}

This program will perform CRUD as well as searching and serialization on ProductModel class.

**Explanation**

* List<ProductModel> pro1 = new List<ProductModel>();

Here, pro1 is list of type ProductModel that means it can store multiple object of ProductModel inside it.

* + **Insert**

Console.WriteLine("Add Product Id");

ID = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Add Product Name");

NAME = Console.ReadLine();

Console.WriteLine("Add Product Category");

CAT = Console.ReadLine();

Console.WriteLine("Add Product price");

PRICE = Convert.ToInt16(Console.ReadLine());

Console.WriteLine("Add Product Qty");

QTY = Convert.ToInt16(Console.ReadLine());

TOTALAMT = PRICE \* QTY;

ProductModel p1 = new ProductModel(ID, NAME, CAT, PRICE, QTY,TOTALAMT);

pro1.Add(p1);

In case 1(Insert) we have created p1 which will call parametrized constructor of ProductModel where we assign user entered value to private member. Once object is created, object is added to list by pro1.Add(p1);

* + **Update**

foreach (ProductModel pro in pro1)

{

if (pro.ProductId == uid)

{

pro.ProductName = UNAME;

pro.Category = UCAT;

pro.Price = UPRICE;

pro.Qty = UQTY;

pro.TotalAmount = UTOT;

}

}

In case 3(edit) we have loop through pro(ProductModel list) and if productid matche to user entered id then we have updated data.

* + **Serialization**

BinaryFormatter bf = new BinaryFormatter();

FileStream fsout = new FileStream("ProductBinary.binary", FileMode.Create, FileAccess.Write);

try

{

using (fsout)

{

bf.Serialize(fsout, pro1);

Console.WriteLine("Serialize");

}

}

catch

{

Console.WriteLine("Not Serialize");

}

In case 6(serialization) we have used binary serialization. For binary serialization we must add System.Runtime.Serialization.Formatters.Binary namespace.

First we have created object of binary formatter which is named as bf, after that we have created object of filestream which is named as fout and it will create fill named as ‘ProductBinary.binary’. If you don’t set path then fill will be stored at default path.

If you have any doubt feel free to ask in comment section.

Happy coding and wish you had a great day!!!