Using Task Parallel Library

Exercise 1: Using Synchronization.

In this exercise we will see how to use TPL for Synchronization.

Task 1: Open VS2012 and create a new Application (WPF/CS/WinForm), name it as 'WPF_Task_SYnchronization'. In the bin/debug folder add the new xml file, name it as 'Products.xml' and add the below XML in it:

```
<?xml version="1.0" encoding="utf-8" ?>
<Products>
  <Product>
    <ProductId>1</ProductId>
    <ProductName>Laptop</ProductName>
    <Category>Electornics</Category>
    <Price>700000</Price>
  </Product>
  <Product>
    <ProductId>2</ProductId>
    <ProductName>Router</ProductName>
    <Category>Electornics</Category>
    <Price>7000</Price>
  </Product>
  <Product>
    <ProductId>3</ProductId>
    <ProductName>Printer</ProductName>
    <Category>Electornics</Category>
    <Price>5000</Price>
  </Product>
  <Product>
    <ProductId>4</ProductId>
    <ProductName>Oil</ProductName>
    <Category>Food</Category>
    <Price>700</Price>
  </Product>
  <Product>
    <ProductId>5</ProductId>
    <ProductName>Vegitables</ProductName>
    <Category>Foods</Category>
    <Price>200</Price>
  </Product>
  <Product>
    <ProductId>6</ProductId>
    <ProductName>Washing Machine</ProductName>
    <Category>Home Appliances</Category>
    <Price>6000</Price>
  </Product>
  <Product>
    <ProductId>7</ProductId>
    <ProductName>Cooker</ProductName>
    <Category>Home Appliances</Category>
    <Price>3000</Price>
  </Product>
  <Product>
    <ProductId>8</ProductId>
    <ProductName>Iron</ProductName>
    <Category>Home Appliances</Category>
    <Price>7500</Price>
  </Product>
  <Product>
```

```
    <ProductId>9</ProductId>
    <ProductName>Laptop</ProductName>
    <Category>Electornics</Category>
    <Price>700000</Price>
  </Product>
  <Product>
    <ProductId>1</ProductId>
    <ProductName>Jeans</ProductName>
    <Category>Fashoin</Category>
    <Price>2000</Price>
  </Product>
  <Product>
    <ProductId>10</ProductId>
    <ProductName>TV</ProductName>
    <Category>Home Appliances</Category>
    <Price>10000</Price>
  </Product>
</Products>
```

Task 2: If you use WPF or WinForm, add a button on the UI and the DataGridView (WinForm), DataGrid (WPF).

Task 3: Add the below class in the project:

```
public class Product
    {
        public int ProductId { get; set; }
        public string ProductName { get; set; }
        public string Category { get; set; }
        public int Price { get; set; }
    }
```

Task 4: Add the below code in the Click event of the button (if console Application add the below code in the Main Method):

```
   CancellationToken ct = new CancellationToken ();

            Task taskGetProducts =
Task.Factory.StartNew<ObservableCollection<Product>>((obj) =>
            {
                ObservableCollection<Product> products = new
ObservableCollection<Product>();
                FileStream Fs = new FileStream("Products.xml",
FileMode.Open);
                XDocument xdoc = XDocument.Load(Fs);
                var prods = from p in
xdoc.Descendants("Products").Descendants("Product")
                            select new Product()
                            {
                                ProductId =
Convert.ToInt32(p.Descendants("ProductId").First().Value),
                                ProductName =
p.Descendants("ProductName").First().Value,
                                Category =
p.Descendants("Category").First().Value,
                                Price =
Convert.ToInt32(p.Descendants("Price").First().Value)
```

```
                                };
                foreach (var item in prods)
                {
                    products.Add(item);
                }
                Fs.Close();
                Fs.Dispose();
                return products;
            },CancellationToken.None).ContinueWith(pd =>
            {
                lstproducts.ItemsSource = pd.Result;
            },TaskScheduler.FromCurrentSynchronizationContext());
//Synchronise the data with the ListBox on UI Thread
        }
```

The above code load the XML file and read data from it. The ContinuouWith method synchronizes the data with theDataGrid on the UI of name 'lstproducts'

Run the application and click on the button, the data will be displayed in the DataGrid.

Exercise 2: Using Task class for performing Parallel Operations

Task 1 : Create a new WPF application, name it as 'WPF_Using_Task_Class'.

Task 2: Open IIS and create 3 Web Application of name Server 1, Server 2 and Server 3. Add images in each application.

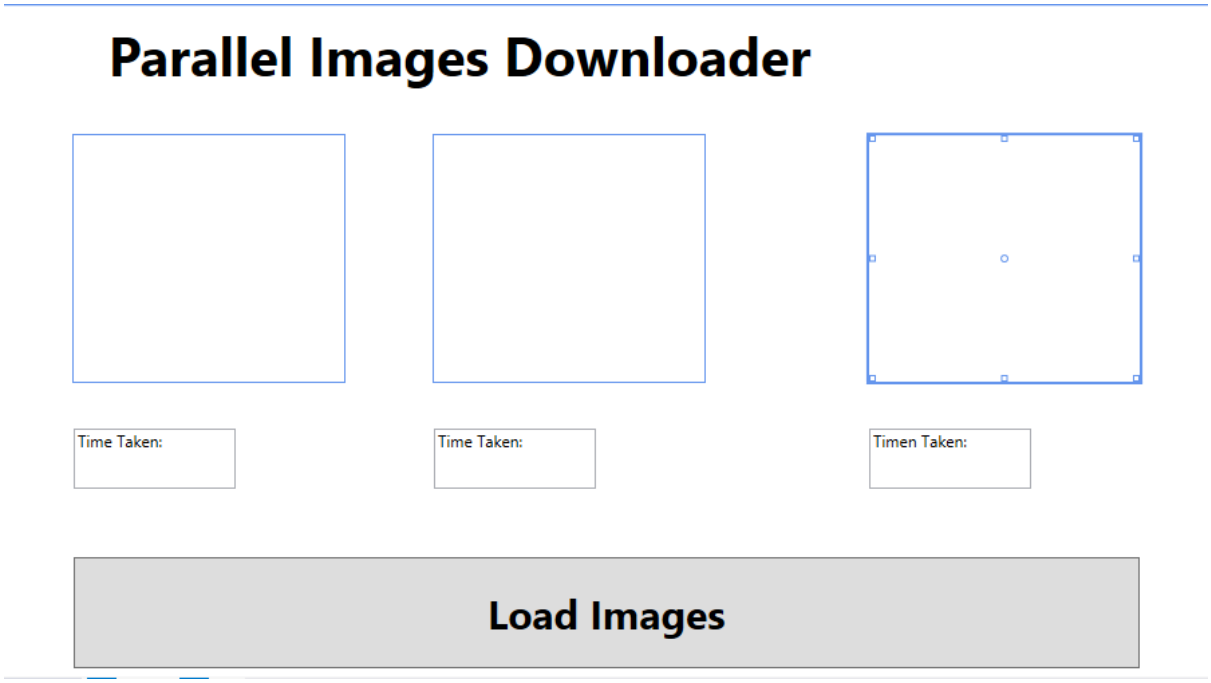
The URL of mages will be as below:

http://localhost/Server1/Chart1.png

http://localhost/Server2/Chart2.png

http://localhost/Server3/Chart3.png

Task 3: Design the UI as below:



It has 3 images, 3 textblocks and a button. (Note: If you are using ASP.NET the instead of the TextBlock use Lables)

Task 4: Add the below code in the code behind, this code is used to generate image from memory stream (Note: This code is specific to WPF, for other clients e.g. ASP.NET the logic will be different).

```
/// <summary>
        /// The helper method to get the BitmapImage from Stream
        /// </summary>
        /// <param name="ms"></param>
        /// <returns></returns>
        ImageSource GetImageFromStream(MemoryStream ms)
        {
            var imageSource = new BitmapImage();
            imageSource.BeginInit();
            imageSource.StreamSource = ms;
            imageSource.EndInit();
            return imageSource;
        }
```

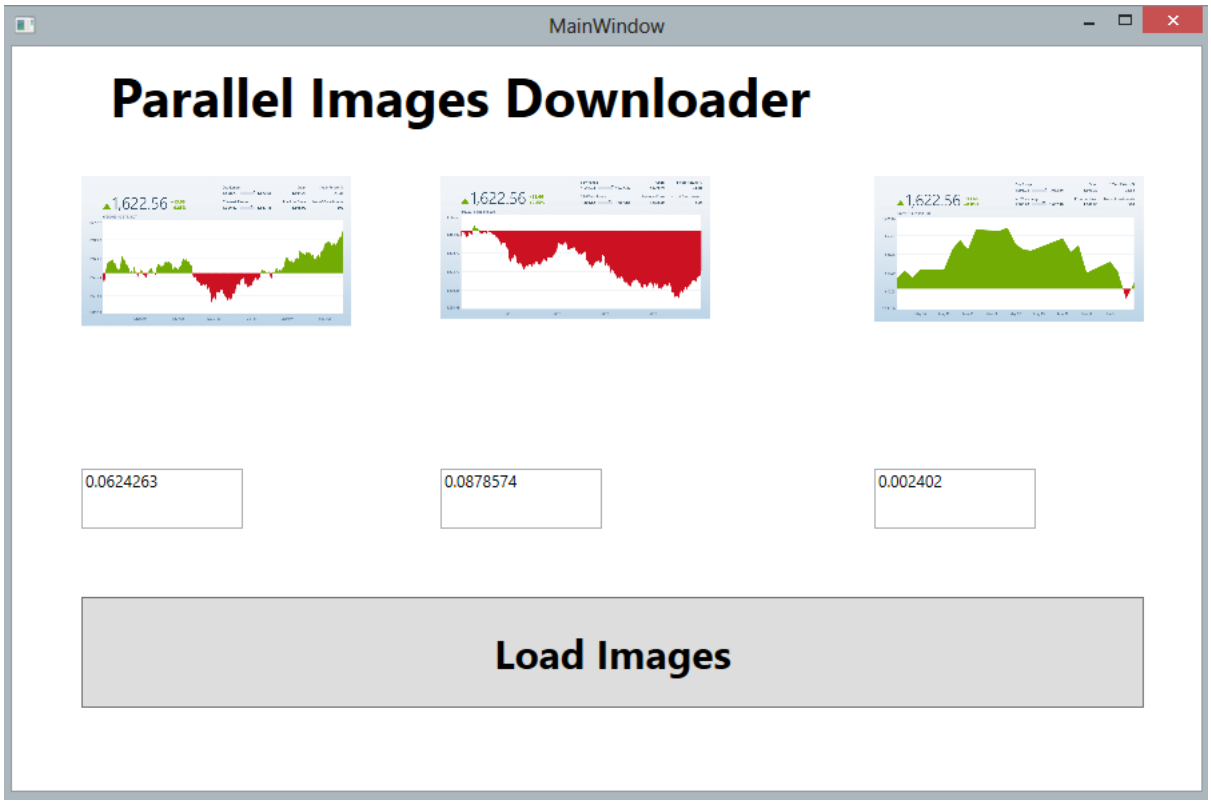
Task 5: Add the below code in the click event of the button:

```
/// <summary>
        /// The method create an object of the Task class and using the
each task is decicated
        /// for downloading the Image from the IIS server
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private  void btnloadimages_Click(object sender, RoutedEventArgs
e)
        {
            try
            {
                string imageServerUri1 =
@"http://localhost/Server1/Chart1.png";
```

```
                string imageServerUri2 =
@"http://localhost/Server2/Chart2.png";
                string imageServerUri3 =
@"http://localhost/Server3/Chart3.png";

                var sw1 = Stopwatch.StartNew();

                Task<Stream> img1 = Task.Factory.StartNew<Stream>(() =>
                    {
                        using (WebClient webClient1 = new WebClient())
                        {
                            return webClient1.OpenRead(new
Uri(imageServerUri1));
                        }
                    });

                MemoryStream ms1 = new MemoryStream();
                img1.Result.CopyTo(ms1);



                // Assign the Source property of your image
               dwimg1.Source = GetImageFromStream(ms1);

                txttime1.Text = sw1.Elapsed.TotalSeconds.ToString();


                var sw2 = Stopwatch.StartNew();

                Task<Stream> img2 = Task.Factory.StartNew<Stream>(() =>
                {
                    using (WebClient webClient2 = new WebClient())
                    {
                        return webClient2.OpenRead(new
Uri(imageServerUri2));
                    }
                });

                MemoryStream ms2 = new MemoryStream();
                img2.Result.CopyTo(ms2);
                // Assign the Source property of your image
                dwimg2.Source = GetImageFromStream(ms2);

                txttime2.Text = sw2.Elapsed.TotalSeconds.ToString();

                var sw3 = Stopwatch.StartNew();

                Task<Stream> img3 = Task.Factory.StartNew<Stream>(() =>
                {
                    using (WebClient webClient3 = new WebClient())
                    {
                        return webClient3.OpenRead(new
Uri(imageServerUri3));
                    }
                });
                MemoryStream ms3 = new MemoryStream();
                img3.Result.CopyTo(ms3);
                // Assign the Source property of your image
                dwimg3.Source = GetImageFromStream(ms3);
                txttime3.Text = sw3.Elapsed.TotalSeconds.ToString();
```

```
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

Run the application and click on the button, you will get the result as below:



Exercise 3: Using Parallel.Invoke

Task 1: Open VS2012 and create a new application add, name it as 'WPF_Parallel_Invoke'. In the bin/debug folder add tow xml files name them as Capacity.xml and Stocks.xml.

Capacity.xml

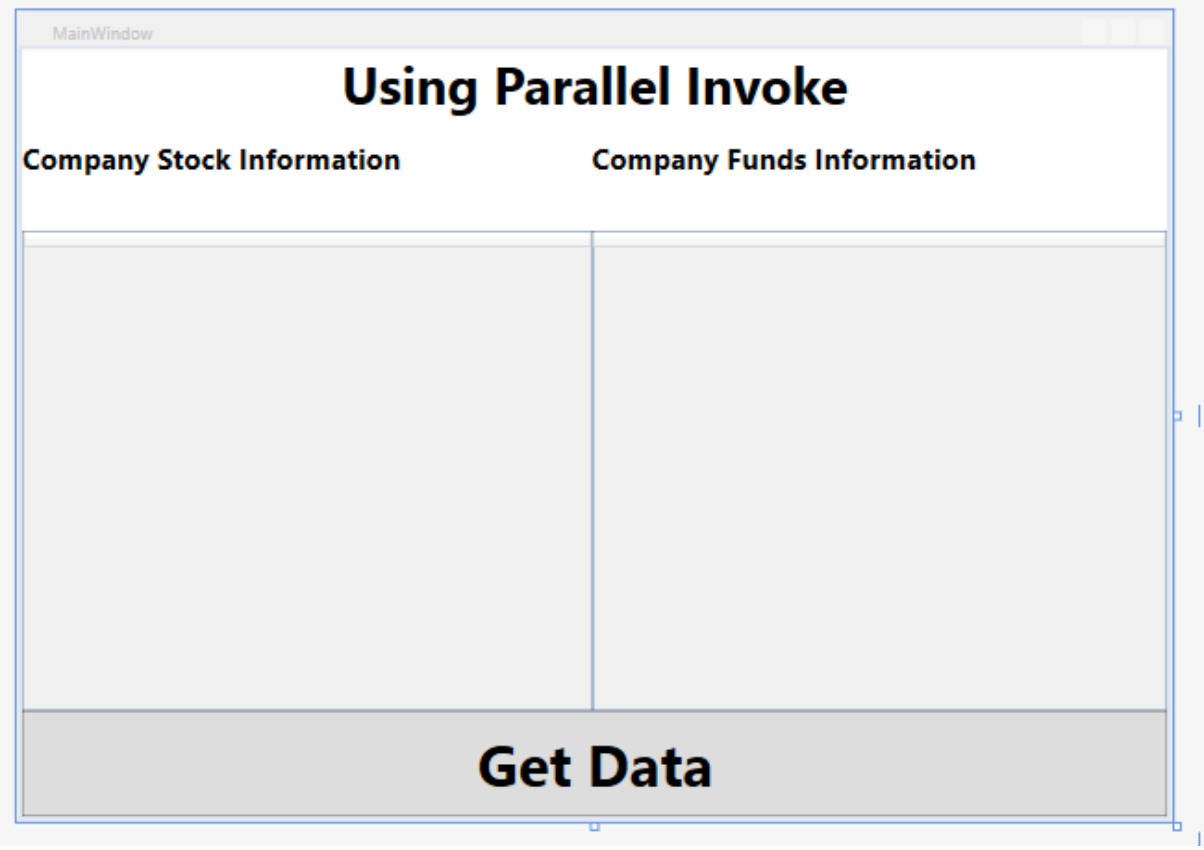
```
<?xml version="1.0" encoding="utf-8" ?>
<FundCapacity>
  <CompanyDetails>
    <Name>MS</Name>
    <Funds>10000</Funds>
  </CompanyDetails>
  <CompanyDetails>
    <Name>LS</Name>
    <Funds>12000</Funds>
  </CompanyDetails>
  <CompanyDetails>
    <Name>TS</Name>
    <Funds>9000</Funds>
  </CompanyDetails>
  <CompanyDetails>
```

```
    <Name>NS</Name>
    <Funds>40000</Funds>
  </CompanyDetails>
  <CompanyDetails>
    <Name>PS</Name>
    <Funds>3000</Funds>
  </CompanyDetails>
</FundCapacity>
```

Stocks.sml

```
<?xml version="1.0" encoding="utf-8" ?>
<StockDetails>
  <CompanyProfile>
    <Name>MS</Name>
    <Stock>1000</Stock>
  </CompanyProfile>
  <CompanyProfile>
    <Name>LS</Name>
    <Stock>1200</Stock>
  </CompanyProfile>
  <CompanyProfile>
    <Name>TS</Name>
    <Stock>2000</Stock>
  </CompanyProfile>
  <CompanyProfile>
    <Name>NS</Name>
    <Stock>3000</Stock>
  </CompanyProfile>
  <CompanyProfile>
    <Name>PS</Name>
    <Stock>2000</Stock>
  </CompanyProfile>
</StockDetails>
```

Task 2: On the UI add two DataGrids and Button as below:



Task 3: Add a new class file on the project name it as DataAccess.cs and add the beolw code in it:

```
  public class Stock
    {
        public string CompanyName { get; set; }
        public int StockQuantity { get; set; }
    }
    public class Fund
    {
        public string CompanyName { get; set; }
        public int Funds { get; set; }
    }

    public static class StockFunds
    {
        /// <summary>
        /// Method to read stock information from Stocks.xml
        /// </summary>
        /// <returns></returns>
        public static ObservableCollection<Stock> GetStocks()
        {
            Debug.WriteLine("In the GetStocks Operation");
            ObservableCollection<Stock> Stocks = new
ObservableCollection<Stock>();
            FileStream Fs = new FileStream("Stocks.xml",FileMode.Open);
            XDocument xdoc = XDocument.Load(Fs);
            var stockData = from s in
xdoc.Descendants("StockDetails").Descendants("CompanyProfile")
                            select new Stock()
                            {
                                CompanyName =
s.Descendants("Name").First().Value,
```

```
                                    StockQuantity =
Convert.ToInt32(s.Descendants("Stock").First().Value)
                                };

            foreach (var item in stockData)
            {
                Stocks.Add(item);
            }
            Fs.Close();
            Fs.Dispose();
            return Stocks;
        }

        /// <summary>
        /// Method to read funds from Capacity.xml
        /// </summary>
        /// <returns></returns>
        public static ObservableCollection<Fund> GetFunds()
        {
            Debug.WriteLine("In the GetFunds Operation");
            ObservableCollection<Fund> Funds = new
ObservableCollection<Fund>();
            FileStream Fs = new FileStream("Capacity.xml",FileMode.Open);
            XDocument xdoc = XDocument.Load(Fs);
            var fundData = from f in
xdoc.Descendants("FundCapacity").Descendants("CompanyDetails")
                           select new Fund()
                           {
                                CompanyName =
f.Descendants("Name").First().Value,
                                Funds =
Convert.ToInt32(f.Descendants("Funds").First().Value)
                           };
            foreach (var item in fundData)
            {
                Funds.Add(item);
            }
            Fs.Close();
            Fs.Dispose();
            return Funds;
        }
    }
```

Task 4: Add the below code on the click event of the button:

```
  private void btngetdata_Click(object sender, RoutedEventArgs e)
        {

            ObservableCollection<Stock> Stocks = new
ObservableCollection<Stock>();
            ObservableCollection<Fund> Funds = new
ObservableCollection<Fund>();

            //Make the Parallel call to these methdos
            Parallel.Invoke(() =>
                {
```

```
                    Stocks = StockFunds.GetStocks();
                    Funds = StockFunds.GetFunds();
                }
            );
            dgstocks.ItemsSource = Stocks;
            dgfunds.ItemsSource = Funds;

        }
```

The above code load the XML file and read data from them.

Run the application, you will get the data displayed in both DataGrids.