CollectionView for WinForms

2019.04.09 更新

グレープシティ株式会社

目次

CollectionView

CollectionView for WinForms is a cross platform library that acts as a powerful tool for organizing data and provides formatted data that can be bound to a data-aware control. Simply put, CollectionView simplifies interaction with the data by wrapping around the data for supporting operations, such as filtering and sorting, which aren't commonly supported by data collections. To achieve this, the library offers C1CollectionView class that implements the ICollectionView interface for supporting current record management, filtering, grouping and sorting operations for your data collection. The ICollectionView interface is the primary data view object and essentially a view of the underlying data source that allows you to manipulate your data without actually modifying the underlying values. CollectionView is designed to be used with data controls, such as DataGridView and Chart and allows you to define your own rules for performing filtering, grouping, etc.

主な特長

CollectionView offers many advanced features beyond simple data management. These features are listed below:

- **Powerful Library**
  CollectionView is a powerful and flexible view that is designed to perform common data transformations by implementing the ICollectionView interface.

- **Optimized Performance**
  CollectionView is highly optimized in performing operations such as, sorting, filtering, and grouping on large data sets efficiently.

- **Manipulate Data with Sorting, Filtering, Grouping and Record Management**
  CollectionView class and ICollectionView interface provides support for data manipulation in the form of sorting, filtering, record management, and grouping collections in WinForms apps.

- **Easy to use with Any Control**
  CollectionView is a compatible data source that can be used with any data-aware control, such as DataGridView, DataFilter and FlexGrid.

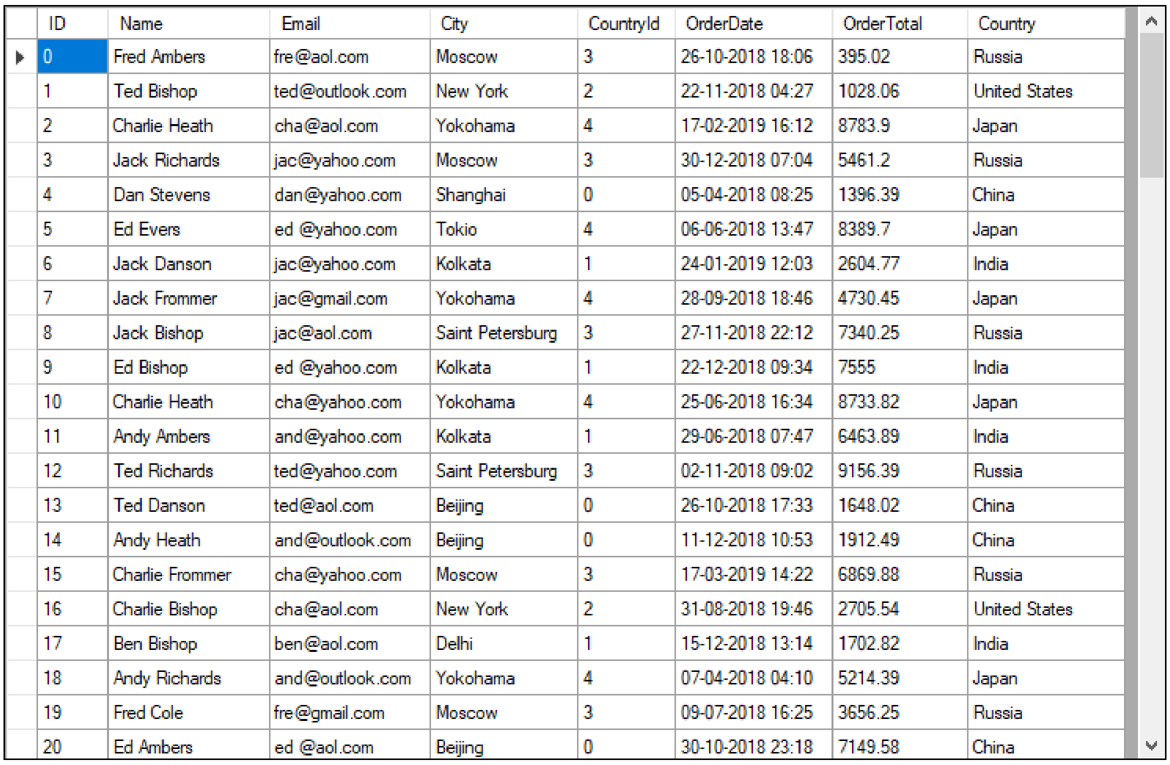
クイックスタート

This quick start will guide you through the steps of adding a DataGridView control to your application and add data to it using the C1CollectionView class.

Complete the steps given below to see how the DataGridView control appears after data binding:

1. **Create a data source**
2. **Bind DataGridView to the data source**

The following image shows how the DataGridView control appears after completing the steps above.



ID	Name	Email	City	CountryId	OrderDate	OrderTotal	Country
0	Fred Ambers	fre@aol.com	Moscow	3	26-10-2018 18:06	395.02	Russia
1	Ted Bishop	ted@outlook.com	New York	2	22-11-2018 04:27	1028.06	United States
2	Charlie Heath	cha@aol.com	Yokohama	4	17-02-2019 16:12	8783.9	Japan
3	Jack Richards	jac@yahoo.com	Moscow	3	30-12-2018 07:04	5461.2	Russia
4	Dan Stevens	dan@yahoo.com	Shanghai	0	05-04-2018 08:25	1396.39	China
5	Ed Evers	ed @yahoo.com	Tokio	4	06-06-2018 13:47	8389.7	Japan
6	Jack Danson	jac@yahoo.com	Kolkata	1	24-01-2019 12:03	2604.77	India
7	Jack Frommer	jac@gmail.com	Yokohama	4	28-09-2018 18:46	4730.45	Japan
8	Jack Bishop	jac@aol.com	Saint Petersburg	3	27-11-2018 22:12	7340.25	Russia
9	Ed Bishop	ed @yahoo.com	Kolkata	1	22-12-2018 09:34	7555	India
10	Charlie Heath	cha@yahoo.com	Yokohama	4	25-06-2018 16:34	8733.82	Japan
11	Andy Ambers	and@yahoo.com	Kolkata	1	29-06-2018 07:47	6463.89	India
12	Ted Richards	ted@yahoo.com	Saint Petersburg	3	02-11-2018 09:02	9156.39	Russia
13	Ted Danson	ted@aol.com	Beijing	0	26-10-2018 17:33	1648.02	China
14	Andy Heath	and@outlook.com	Beijing	0	11-12-2018 10:53	1912.49	China
15	Charlie Frommer	cha@yahoo.com	Moscow	3	17-03-2019 14:22	6869.88	Russia
16	Charlie Bishop	cha@aol.com	New York	2	31-08-2018 19:46	2705.54	United States
17	Ben Bishop	ben@aol.com	Delhi	1	15-12-2018 13:14	1702.82	India
18	Andy Richards	and@outlook.com	Yokohama	4	07-04-2018 04:10	5214.39	Japan
19	Fred Cole	fre@gmail.com	Moscow	3	09-07-2018 16:25	3656.25	Russia
20	Ed Ambers	ed @aol.com	Beijing	0	30-10-2018 23:18	7149.58	China

Step 1: Create a data source

1. Add a new class file, Customer, to the application.
2. Add the following code to the `Customer` file. In this example, we are using **Customer** class to represent data in the DataGridView control.

Visual Basic

```
Public Class Customer
    Private _id, _countryId As Integer
    Private _name, _email, _city As String
    Private _OrderDate As DateTime
    Private _orderTotal As Double

    Shared _rnd As Random = New Random()
    Shared _firstNames As String() =
"Andy|Ben|Charlie|Dan|Ed|Fred|Herb|Jack|Mark|Ted".Split("|"c)
    Shared _lastNames As String() =
"Ambers|Bishop|Cole|Danson|Evers|Frommer|Heath|Myers|Richards|Stevens".Split("|"c)

    Shared _emailServers As String() = "gmail|yahoo|outlook|aol".Split("|"c)
    Shared countries As String =
       "China-Beijing,Shanghai|India-Delhi,Kolkata|United States-Washington,New
York|Russia-Moscow,Saint Petersburg|Japan-Tokio,Yokohama"
    Shared _countries As KeyValuePair(Of String, String())() =
       countries.Split("|"c).[Select](Function(str) New KeyValuePair(Of String,
String())(str.Split("-"c).First(), str.Split("-
"c).Skip(1).First().Split(","c))).ToArray()

    Public Sub New()
    End Sub

    Public Sub New(ByVal id As Integer)
        ID = id
        Name = GetName()
        Email = String.Format("{0}@{1}.com", (Name.Substring(0, 3)).ToLower(),
GetString(_emailServers))
        CountryId = _rnd.[Next]() Mod _countries.Length
        Dim cities = _countries(CountryId).Value
        City = GetString(cities)
        OrderDate = DateTime.Today.AddDays(-_rnd.[Next](1, 365)).AddHours(_rnd.
[Next](0, 24)).AddMinutes(_rnd.[Next](0, 60))
        OrderTotal = Math.Round(_rnd.NextDouble() * 10000.00, 2)
    End Sub

    Public Property ID As Integer
        Get
            Return _id
        End Get
        Set(ByVal value As Integer)

            If value <> _id Then
                _id = value
            End If
        End Set
    End Property

    Public Property Name As String
        Get
            Return _name
        End Get
        Set(ByVal value As String)

            If value <> _name Then
                _name = value
            End If
        End Set
    End Property

    Public Property Email As String
        Get
            Return _email
        End Get
        Set(ByVal value As String)

            If value <> _email Then
                _email = value
            End If
        End Set
    End Property

    Public Property City As String
        Get
            Return _city
        End Get
        Set(ByVal value As String)
```

```
            If value <> _city Then
                _city = value
            End If
        End Set
    End Property

    Public Property CountryId As Integer
        Get
            Return _countryId
        End Get
        Set(ByVal value As Integer)

            If value <> _countryId AndAlso value > -1 AndAlso value <
_countries.Length Then
                _countryId = value
            End If
        End Set
    End Property

    Public Property OrderDate As DateTime
        Get
            Return _OrderDate
        End Get
        Set(ByVal value As DateTime)

            If value <> _OrderDate Then
                _OrderDate = value
            End If
        End Set
    End Property

    Public Property OrderTotal As Double
        Get
            Return _orderTotal
        End Get
        Set(ByVal value As Double)

            If value <> _orderTotal Then
                _orderTotal = value
            End If
        End Set
    End Property

    Private Shared Function GetString(ByVal arr As String()) As String
        Return arr(_rnd.[Next](arr.Length))
    End Function

    Private Shared Function GetName() As String
        Return String.Format("{0} {1}", GetString(_firstNames),
GetString(_lastNames))
    End Function

    Public ReadOnly Property Country As String
        Get
            Return _countries(_countryId).Key
        End Get
    End Property

    Public Shared Function GetCustomerList(ByVal count As Integer) As
ObservableCollection(Of Customer)
        Dim list = New ObservableCollection(Of Customer)()
        For i As Integer = 0 To count - 1
            list.Add(New Customer(i))
        Next

        Return list
    End Function
End Class
```

C#

```
public class Customer
{
   int _id, _countryId;
   string _name, _email, _city;
   DateTime _OrderDate;
   double _orderTotal;

   static Random _rnd = new Random();
   static string[] _firstNames =
      "Andy|Ben|Charlie|Dan|Ed|Fred|Herb|Jack|Mark|Ted".Split('|');
   static string[] _lastNames =

"Ambers|Bishop|Cole|Danson|Evers|Frommer|Heath|Myers|Richards|Stevens".Split('|');

   static string[] _emailServers = "gmail|yahoo|outlook|aol".Split('|');
   static string countries =
      "China-Beijing,Shanghai|India-Delhi,Kolkata|United States-Washington,New
York|Russia-Moscow,Saint Petersburg|Japan-Tokio,Yokohama";
   static KeyValuePair<string, string[]>[] _countries =
      countries.Split('|').Select(str => new KeyValuePair<string, string[]>
(str.Split('-').First(),
          str.Split('-').Skip(1).First().Split(','))).ToArray();


   public Customer()
   {
   }

   public Customer(int id)
   {
      ID = id;
      Name = GetName();
      Email = string.Format("{0}@{1}.com", (Name.Substring(0, 3)).ToLower(),
GetString(_emailServers));
      CountryId = _rnd.Next() % _countries.Length;
      var cities = _countries[CountryId].Value;
      City = GetString(cities);
      OrderDate = DateTime.Today.AddDays(-_rnd.Next(1,
365)).AddHours(_rnd.Next(0, 24)).AddMinutes(_rnd.Next(0, 60));
      OrderTotal = Math.Round(_rnd.NextDouble() * 10000.00, 2);
   }

   public int ID
   {
      get { return _id; }
      set
      {
         if (value != _id)
         {
            _id = value;
         }
```

```
        }
    }
    public string Name
    {
        get { return _name; }
        set
        {
            if (value != _name)
            {
                _name = value;
            }
        }
    }
    public string Email
    {
        get { return _email; }
        set
        {
            if (value != _email)
            {
                _email = value;
            }
        }
    }
    public string City
    {
        get { return _city; }
        set
        {
            if (value != _city)
            {
                _city = value;
            }
        }
    }

    public int CountryId
    {
        get { return _countryId; }
        set
        {
            if (value != _countryId && value > -1 && value < _countries.Length)
            {
                _countryId = value;
            }
          }
    }
    public DateTime OrderDate
    {
        get { return _OrderDate; }
        set
        {
            if (value != _OrderDate)
            {
                _OrderDate = value;
            }
        }
    }

    public double OrderTotal
    {
        get { return _orderTotal; }
        set
        {
            if (value != _orderTotal)
            {
                 _orderTotal = value;
            }
        }
    }

    // ** utilities
    static string GetString(string[] arr)
    {
         return arr[_rnd.Next(arr.Length)];
    }
    static string GetName()
    {
         return string.Format("{0} {1}", GetString(_firstNames),
GetString(_lastNames));
    }
    public string Country
    {
         get { return _countries[_countryId].Key; }
    }

    // ** static list provider
    public static ObservableCollection<Customer> GetCustomerList(int count)
    {
         var list = new ObservableCollection<Customer>();
         for (int i = 0; i < count; i++)
         {
             list.Add(new Customer(i));
         }
         return list;
     }
}
```

Back to Top

Step 2: Bind DataGridView to the data source

1. Add the following dlls to your application to work with CollectionView:
    - C1.CollectionView.dll
    - C1.Win.CollectionView.dll

   You can also use the available CollectionView NuGet packages from the following locations:

    - C1.CollectionView: https://www.nuget.org/packages/C1.CollectionView
    - C1.Win.CollectionView: https://www.nuget.org/packages/C1.Win.CollectionView

   For information on how to add NuGet packages to your application, see **Adding NuGet Packages to your App**.

2. Drag and drop the DataGridView control from the Toolbox onto your form.
3. Switch to the Code view and add the following code to bind DataGridView to the data source.
    - **Visual Basic**

```
Dim cv As C1CollectionView(Of Customer) =
        New C1CollectionView(Of Customer)(Customer.GetCustomerList(100))
gridview.DataSource = New C1CollectionViewBindingList(cv)
```

    - **C#**

```
C1CollectionView<Customer> cv = new C1CollectionView<Customer>(Customer.GetCustomerList(100));
gridview.DataSource = new C1CollectionViewBindingList(cv);
```

Run the application and observe that the grid displays a Customers table.

Back to Top

CollectionView の操作

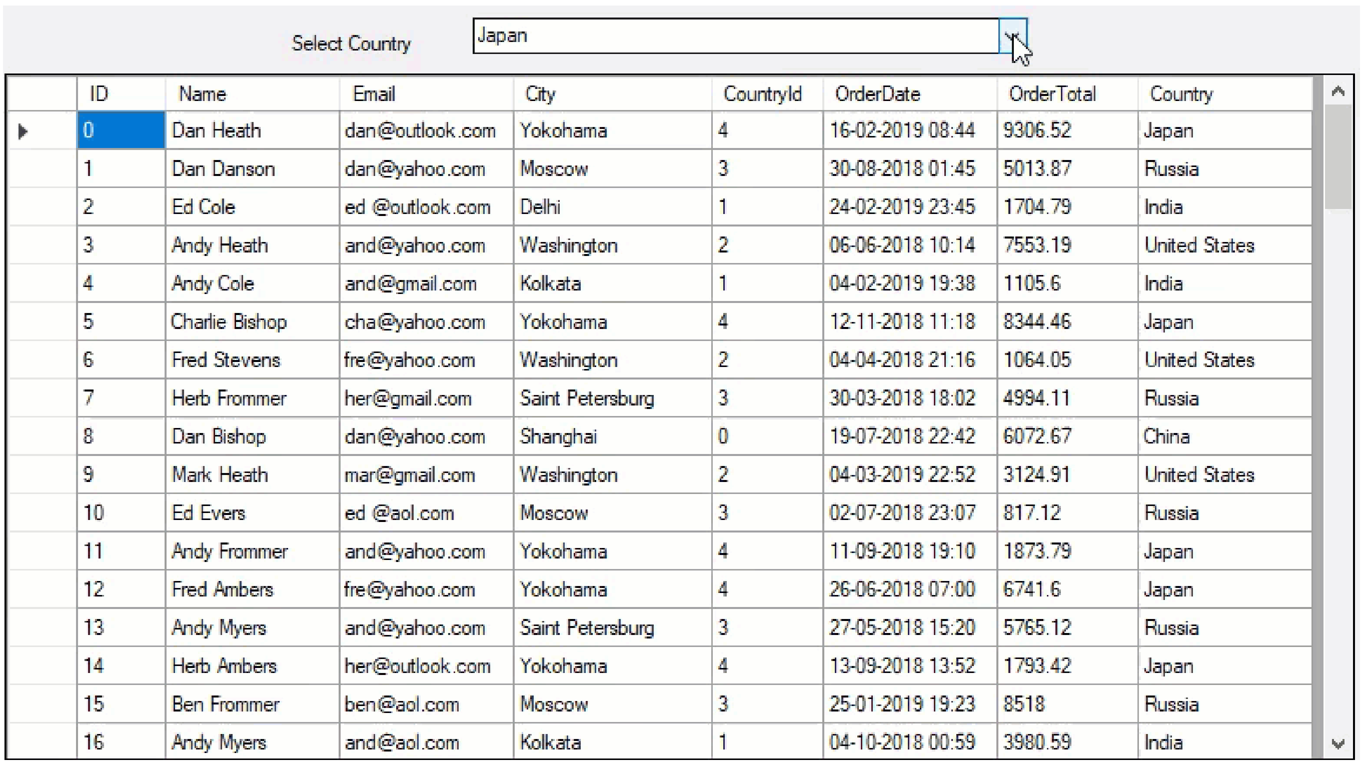
This section comprises all the functionalities of CollectionView.

現在のレコード管理

CollectionView manages current record by implementing the **ICollectionView** interface. It allows you to obtain the current position of a record in the collection using **CurrentPosition** property of the **C1CollectionView** class. This class provides various methods to change the current position of an item in a view, which are listed below:

- **MoveCurrentTo** - Sets the specified item to be the current item in the view.
- **MoveCurrentToFirst** - Sets the first item in the view as the current item.
- **MoveCurrentToLast** - Sets the last item in the view as the current item.
- **MoveCurrentToNext** - Sets the item after the current item in the view as the current item.
- **MoveCurrentToPosition(int)** - Sets the item at the specified index in the view as current item.
- **MoveCurrentToPrevious** - Sets the item before the current item in the view as the current item.

The following GIF displays how the current record management is implemented using MoveCurrentTo method.



Select Country: Japan

	ID	Name	Email	City	CountryId	OrderDate	OrderTotal	Country
▶	0	Dan Heath	dan@outlook.com	Yokohama	4	16-02-2019 08:44	9306.52	Japan
	1	Dan Danson	dan@yahoo.com	Moscow	3	30-08-2018 01:45	5013.87	Russia
	2	Ed Cole	ed @outlook.com	Delhi	1	24-02-2019 23:45	1704.79	India
	3	Andy Heath	and@yahoo.com	Washington	2	06-06-2018 10:14	7553.19	United States
	4	Andy Cole	and@gmail.com	Kolkata	1	04-02-2019 19:38	1105.6	India
	5	Charlie Bishop	cha@yahoo.com	Yokohama	4	12-11-2018 11:18	8344.46	Japan
	6	Fred Stevens	fre@yahoo.com	Washington	2	04-04-2018 21:16	1064.05	United States
	7	Herb Frommer	her@gmail.com	Saint Petersburg	3	30-03-2018 18:02	4994.11	Russia
	8	Dan Bishop	dan@yahoo.com	Shanghai	0	19-07-2018 22:42	6072.67	China
	9	Mark Heath	mar@gmail.com	Washington	2	04-03-2019 22:52	3124.91	United States
	10	Ed Evers	ed @aol.com	Moscow	3	02-07-2018 23:07	817.12	Russia
	11	Andy Frommer	and@yahoo.com	Yokohama	4	11-09-2018 19:10	1873.79	Japan
	12	Fred Ambers	fre@yahoo.com	Yokohama	4	26-06-2018 07:00	6741.6	Japan
	13	Andy Myers	and@yahoo.com	Saint Petersburg	3	27-05-2018 15:20	5765.12	Russia
	14	Herb Ambers	her@outlook.com	Yokohama	4	13-09-2018 13:52	1793.42	Japan
	15	Ben Frommer	ben@aol.com	Moscow	3	25-01-2019 19:23	8518	Russia
	16	Andy Myers	and@aol.com	Kolkata	1	04-10-2018 00:59	3980.59	India

In the following example, we used DataGridView and ComboBox controls wherein item selected from the ComboBox is set as the current item in DataGridView using the **MoveCurrentTo** method. By default, invoking the MoveCurrentTo method sets specified item in the view as the current item. However, you can select and move the current row to the top of the grid by handling the CurrentChanged event as implemented in the code below:

- **Visual Basic**

```
Private cv As C1CollectionView(Of Customer)

Public Sub New()
    cv = New C1CollectionView(Of Customer)(Customer.GetCustomerList(100))
    gridview.DataSource = New C1CollectionViewBindingList(cv)
    ComboBox1.DisplayMember = "Country"
    ComboBox1.DataSource = New C1CollectionViewBindingList(cv)
    cv.CurrentChanged += cv_CurrentChanged()
End Sub

Private Sub cv_CurrentChanged(ByVal sender As Object, ByVal e As EventArgs)
    gridview.FirstDisplayedScrollingRowIndex = cv.CurrentPosition
    gridview.ClearSelection()
    gridview.Rows(cv.CurrentPosition).Selected = True
End Sub

Private Sub ComboBox1_SelectedIndexChanged(sender As Object,
            e As EventArgs) Handles ComboBox1.SelectedIndexChanged
    cv.MoveCurrentTo(ComboBox1.SelectedItem)
End Sub
```

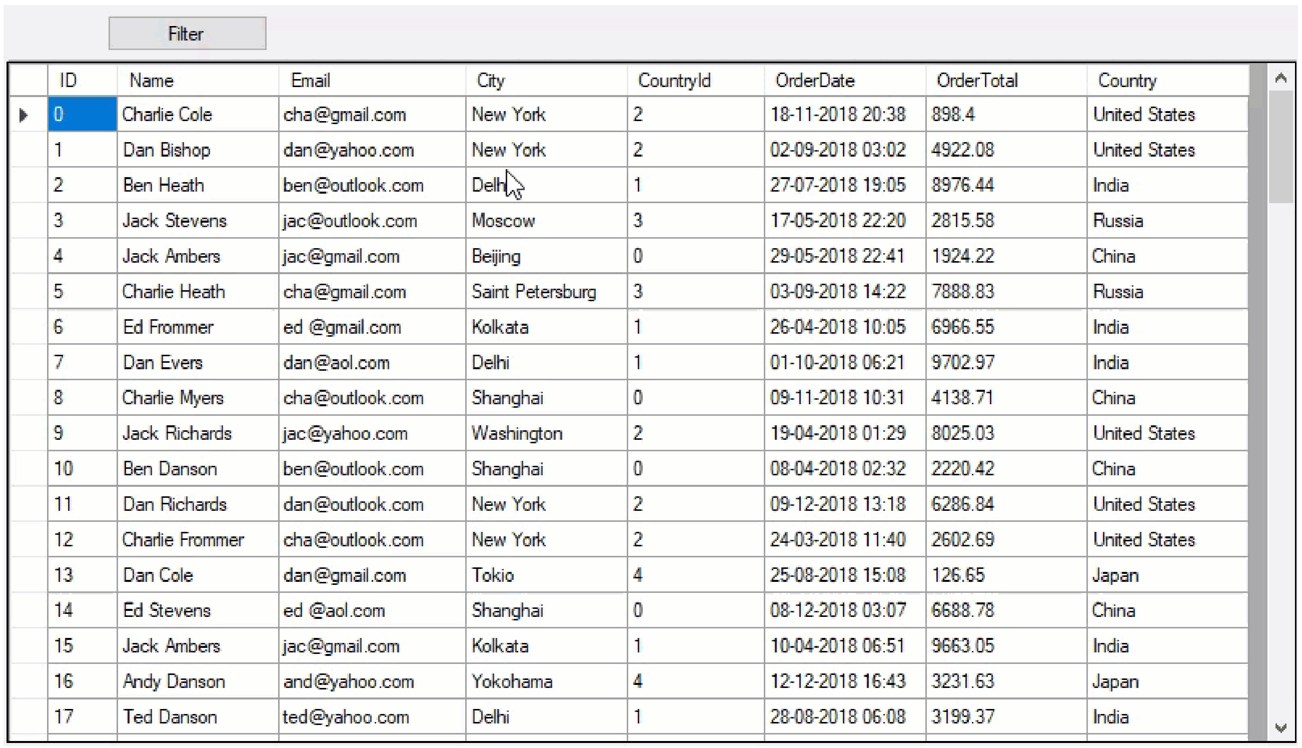
- **C#**

```
C1CollectionView<Customer> cv;
public RecordManagement()
{
    InitializeComponent();

    cv = new C1CollectionView<Customer>(Customer.GetCustomerList(100));
    gridview.DataSource = new C1CollectionViewBindingList(cv);
    //cbCustomerはComboBoxです
    cbCustomer.DisplayMember = "Country";
    cbCustomer.DataSource = new C1CollectionViewBindingList(cv);

    cv.CurrentChanged += cv_CurrentChanged;
}

private void cv_CurrentChanged(object sender, EventArgs e)
{
    gridview.FirstDisplayedScrollingRowIndex = cv.CurrentPosition;
    gridview.ClearSelection();
    gridview.Rows[cv.CurrentPosition].Selected = true;
}

private void cbCustomer_SelectedIndexChanged(object sender, EventArgs e)
{
    cv.MoveCurrentTo(cbCustomer.SelectedItem);
}
```

フィルタ処理

CollectionView implements the **ICollectionView** interface to support filtering, which enables you to filter data using the **FilterAsync** method of the **C1CollectionView** class. This method calls the filtering operation in the collection view and refines data according to the user requirements without including other data that can be repetitive or irrelevant. CollectionView fires **FilterChanged** event when a filter operation is performed. In addition, CollectionView allows you to fetch the filter expression applied to the data using the **FilterExpression** property.

The following GIF displays how the filtering is implemented using the FilterAsync method.



ID	Name	Email	City	CountryId	OrderDate	OrderTotal	Country
0	Charlie Cole	cha@gmail.com	New York	2	18-11-2018 20:38	898.4	United States
1	Dan Bishop	dan@yahoo.com	New York	2	02-09-2018 03:02	4922.08	United States
2	Ben Heath	ben@outlook.com	Delhi	1	27-07-2018 19:05	8976.44	India
3	Jack Stevens	jac@outlook.com	Moscow	3	17-05-2018 22:20	2815.58	Russia
4	Jack Ambers	jac@gmail.com	Beijing	0	29-05-2018 22:41	1924.22	China
5	Charlie Heath	cha@gmail.com	Saint Petersburg	3	03-09-2018 14:22	7888.83	Russia
6	Ed Frommer	ed @gmail.com	Kolkata	1	26-04-2018 10:05	6966.55	India
7	Dan Evers	dan@aol.com	Delhi	1	01-10-2018 06:21	9702.97	India
8	Charlie Myers	cha@outlook.com	Shanghai	0	09-11-2018 10:31	4138.71	China
9	Jack Richards	jac@yahoo.com	Washington	2	19-04-2018 01:29	8025.03	United States
10	Ben Danson	ben@outlook.com	Shanghai	0	08-04-2018 02:32	2220.42	China
11	Dan Richards	dan@outlook.com	New York	2	09-12-2018 13:18	6286.84	United States
12	Charlie Frommer	cha@outlook.com	New York	2	24-03-2018 11:40	2602.69	United States
13	Dan Cole	dan@gmail.com	Tokio	4	25-08-2018 15:08	126.65	Japan
14	Ed Stevens	ed @aol.com	Shanghai	0	08-12-2018 03:07	6688.78	China
15	Jack Ambers	jac@gmail.com	Kolkata	1	10-04-2018 06:51	9663.05	India
16	Andy Danson	and@yahoo.com	Yokohama	4	12-12-2018 16:43	3231.63	Japan
17	Ted Danson	ted@yahoo.com	Delhi	1	28-08-2018 06:08	3199.37	India

The following code implements filtering in DataGridView according to the specified filtering criteria using appropriate filter operator and filter values in the **FilterAsync** method. In this example, data is filtered in DataGridView on the basis of the provided filter condition. This example uses the sample created in the Quick Start section.

- **Visual Basic**

```
Private Async Sub btnFilter_Click(sender As Object, e As EventArgs) Handles btnFilter.Click
    Await cv.FilterAsync("Name", FilterOperation.StartsWith, "He")
End Sub
```

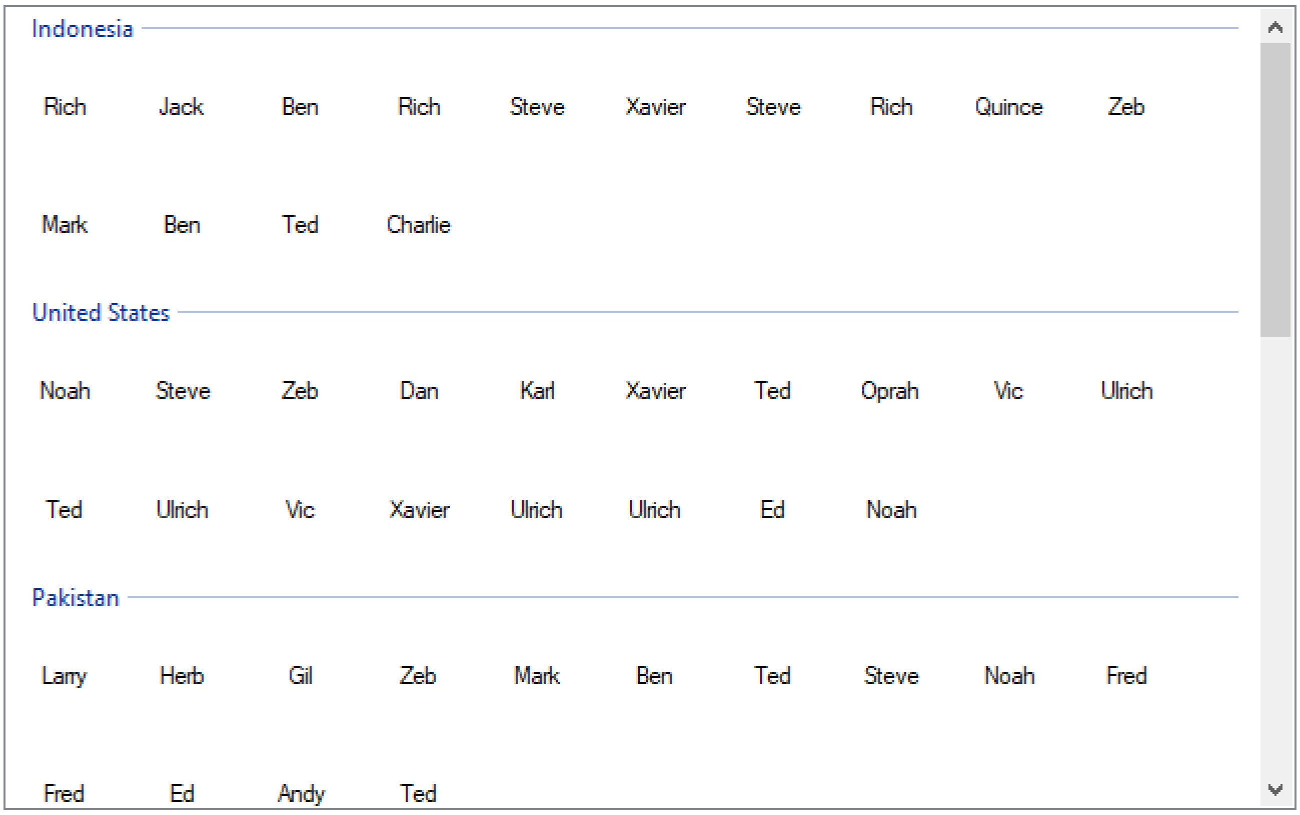
- **C#**

```
private async void btnFilter_Click(object sender, EventArgs e)
{
    await cv.FilterAsync("Name", FilterOperation.StartsWith, "He");
}
```

グループ化

CollectionView implements the **ICollectionView** interface to support grouping, which enables you to group data using the **GroupAsync** method of the **C1CollectionView** class. This method calls the grouping operation in the collection view and groups data according to the specified field names, group path, or group descriptions. When grouping is applied, it automatically sorts the data and splits it into groups by combining rows based on column values.

The following image shows how the customer names are grouped by country in a ListView.



```
Indonesia
Rich    Jack    Ben     Rich    Steve   Xavier  Steve   Rich    Quince  Zeb
Mark    Ben     Ted     Charlie
United States
Noah    Steve   Zeb     Dan     Karl    Xavier  Ted     Oprah   Vic     Ulrich
Ted     Ulrich  Vic     Xavier  Ulrich  Ulrich  Ed      Noah
Pakistan
Larry   Herb    Gil     Zeb     Mark    Ben     Ted     Steve   Noah    Fred
Fred    Ed      Andy    Ted
```

The following code implements grouping in ListView using the **GroupAsync** method. In this example, name of the customers is grouped by country in the ListView control.

- **Visual Basic**

```
Private cv As C1CollectionView(Of Customer)

Public Sub New()
    InitializeComponent()
    cv = New C1CollectionView(Of Customer)(Customer.GetCustomerList(100))
    ListView1.SetItemsSource(cv, "Name")
    ListView1.Visible = True
End Sub
Private Sub btnGroup_Click(sender As Object, e As EventArgs) Handles btnGroup.Click
    ListView1.SetItemsSource(cv, "Name", "City")
    cv.GroupAsync(Function(v) v.Customer)
End Sub
```

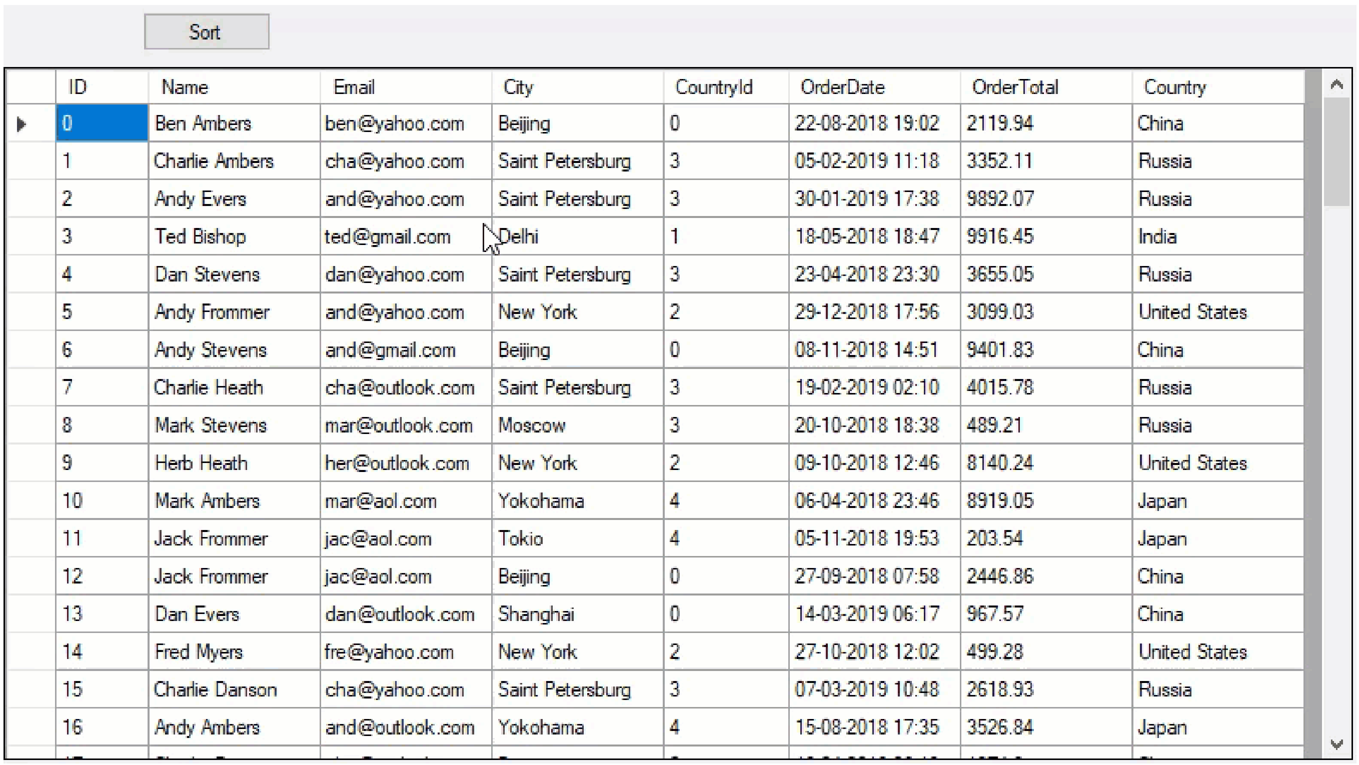
- **C#**

```
C1CollectionView<Customer> cv;
public Grouping()
{
    InitializeComponent();

    cv = new C1CollectionView<Customer>(Customer.GetCustomerList(100));
    listView1.SetItemsSource(cv, "Name");
    listView1.Visible = true;
}

private void btnGroup_Click(object sender, EventArgs e)
{
    listView1.SetItemsSource(cv, "Name", "City");
    cv.GroupAsync(v => v.Country);
}
```

ソート

CollectionView implements the **ICollectionView** interface to support sorting data in ascending and descending order. It enables you to sort data according to the specified sort path and direction using **SortAsync** method of the **C1CollectionView** class. CollectionView also allows you to set the direction of sort operation using **Direction** property of the **SortDescription** class which accepts values from the **SortDirection** enumeration. Moreover, it allows you to specify the path of a data item to which the sort operation needs to be applied using the **SortPath** property.

The following GIF displays how the sorting is implemented using the SortAsync method.



ID	Name	Email	City	CountryId	OrderDate	OrderTotal	Country
0	Ben Ambers	ben@yahoo.com	Beijing	0	22-08-2018 19:02	2119.94	China
1	Charlie Ambers	cha@yahoo.com	Saint Petersburg	3	05-02-2019 11:18	3352.11	Russia
2	Andy Evers	and@yahoo.com	Saint Petersburg	3	30-01-2019 17:38	9892.07	Russia
3	Ted Bishop	ted@gmail.com	Delhi	1	18-05-2018 18:47	9916.45	India
4	Dan Stevens	dan@yahoo.com	Saint Petersburg	3	23-04-2018 23:30	3655.05	Russia
5	Andy Frommer	and@yahoo.com	New York	2	29-12-2018 17:56	3099.03	United States
6	Andy Stevens	and@gmail.com	Beijing	0	08-11-2018 14:51	9401.83	China
7	Charlie Heath	cha@outlook.com	Saint Petersburg	3	19-02-2019 02:10	4015.78	Russia
8	Mark Stevens	mar@outlook.com	Moscow	3	20-10-2018 18:38	489.21	Russia
9	Herb Heath	her@outlook.com	New York	2	09-10-2018 12:46	8140.24	United States
10	Mark Ambers	mar@aol.com	Yokohama	4	06-04-2018 23:46	8919.05	Japan
11	Jack Frommer	jac@aol.com	Tokio	4	05-11-2018 19:53	203.54	Japan
12	Jack Frommer	jac@aol.com	Beijing	0	27-09-2018 07:58	2446.86	China
13	Dan Evers	dan@outlook.com	Shanghai	0	14-03-2019 06:17	967.57	China
14	Fred Myers	fre@yahoo.com	New York	2	27-10-2018 12:02	499.28	United States
15	Charlie Danson	cha@yahoo.com	Saint Petersburg	3	07-03-2019 10:48	2618.93	Russia
16	Andy Ambers	and@outlook.com	Yokohama	4	15-08-2018 17:35	3526.84	Japan

The following code demonstrates the implementation of the **SortAsync** method to sort data in DataGridView. In this example, Name column of DataGridView is sorted alphabetically in ascending order. This example uses the sample created in the Quick Start section.

- **Visual Basic**

```
Private Async Sub btnSort_Click(sender As Object, e As EventArgs) Handles btnSort.Click
    If cv IsNot Nothing Then
        Await cv.SortAsync("Name", SortDirection.Ascending)
    End If
End Sub
```

- **C#**

```
private async void btnSort_Click(object sender, EventArgs e)
{
    if (cv != null)
    {
        await cv.SortAsync("Name", SortDirection.Ascending);
    }
}
```

CollectionView のサンプル

With **C1Studio** installer, you get samples that help you understand the product and its implementation better. CollectionView samples are available in the installed folder - **Documents\ComponentOne Samples\WinForms\CollectionView\CS**.

Sample	Description
Amazon	Includes a sample that demonstrates how to create a custom class to filter products from Amazon by keyword.
C1CollectionView101	Includes a sample that demonstrates sorting, filtering, grouping and current record management using C1CollectionView.