Class that has a number of examples which transform arrays of contents into IEnumerable<IEnumerable<Of Source Type>>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Reflection;

namespace LINQ_Extensions
{
    internal class Nested_IEnumerables
    {
        public Nested_IEnumerables( )
        {
            Examples_Empties( );
            Examples_Values_Arrays( );
            Examples_List_Collections( );
            Examples_Mixed_Array_List( );
            Example_Dictionary( );

        }
        internal void Examples_Empties( )
        {
            // IEnumerable with inner IEnumerable = null
            IEnumerable<IEnumerable<Object>> Example1 = new Object[][]
            {
                null
            };
            Debug.WriteLine(string.Format("{0} Example1",
                MethodInfo.GetCurrentMethod( ).Name));
            CheckCastAndResult(Example1);
            // IEnumerable with inner an Empty array(cast to IEnumerable)
            IEnumerable<IEnumerable<Object>> Example2 = new Object[][]
            {
                new Object[]{}
            };
            Debug.WriteLine(string.Format("{0} Example2",
                MethodInfo.GetCurrentMethod( ).Name));
            CheckCastAndResult(Example2);
            // IEnumerable with inner an Empty IEnumerable
```

```
    IEnumerable<IEnumerable<Object>> Example3 = new Object[][]
    {
        new Object[] {Enumerable.Empty<Object>()}
    };
    Debug.WriteLine(string.Format("{0} Example3",
        MethodInfo.GetCurrentMethod( ).Name));
    CheckCastAndResult(Example3);
}
internal void Examples_Values_Arrays( )
{
    // 2 dimension array initialisation
    IEnumerable<IEnumerable<int>> Example1 = new int[][]
    {
        new int[]{1,2,3},
        new int[]{4,5,6}
    };
    Debug.WriteLine(string.Format("{0} Example",
        MethodInfo.GetCurrentMethod( ).Name));
    CheckCastAndResult(Example1);
}
internal void Examples_List_Collections( )
{
    List<List<int>> List1 = new List<List<int>>( )
    {
        new List<int>() { 1, 2, 3}
    };
    IEnumerable<IEnumerable<int>> Example1 = List1;
    Debug.WriteLine(string.Format("{0} Example1",
        MethodInfo.GetCurrentMethod( ).Name));
    CheckCastAndResult(Example1);
    List<List<int>> List2 = new List<List<int>>( )
    {
        new List<int>() { 1, 2, 3},
        new List<int>(){4,5,6}
    };
    IEnumerable<IEnumerable<int>> Example2 = List2;
    Debug.WriteLine(string.Format("{0} Example2",
        MethodInfo.GetCurrentMethod( ).Name));
    CheckCastAndResult(Example2);
    return;
}
internal void Examples_Mixed_Array_List( )
{
    // Example Used for testing
```

```
    IEnumerable<IEnumerable<Object>> Example = new Object[][]
    {
        (new List<Object>() {1,2,3}).ToArray()
    };
    CheckCastAndResult(Example);
    return;
}

internal void Example_Dictionary( )
{
    Dictionary<int, int> DicTest1 = new Dictionary<int, int>( )
    {
        {1,2}
    };
    List<Dictionary<int, int>> Nested =
        new List<Dictionary<int, int>>( )
    {
        DicTest1
    };
    // Example1 Used for Testing Dictionary to IEnumerable conversion
    IEnumerable<IEnumerable<KeyValuePair<int, int>>> Example1 = Nested;
    Debug.WriteLine(string.Format("{0} Example1",
        MethodInfo.GetCurrentMethod( ).Name));
    CheckCastAndResult(Example1);

    Dictionary<int, List<int>> DicTest2 =
        new Dictionary<int, List<int>>( )
        {
            {1, new List<int>() {1,2,3}}
        };
    // Example2 Used for Demonstrating how to build
    IEnumerable<IEnumerable<int>> Example2 =
        new IEnumerable<int>[]
    {
        Enumerable.Repeat(DicTest2.ElementAt(0).Key, 1)
        .Concat(DicTest2.ElementAt(0).Value)
    };
    Debug.WriteLine(string.Format("{0} Example2",
        MethodInfo.GetCurrentMethod( ).Name));
    CheckCastAndResult(Example2);

    // Example3 Used for demonstrating how to build
    IEnumerable<IEnumerable<int>> Example3 =
        DicTest2.Select(a => a.Value);
```

```
            Debug.WriteLine(string.Format("{0} Example3",
                MethodInfo.GetCurrentMethod( ).Name));
            CheckCastAndResult(Example3);

            Dictionary<string, string> DicTest3 =
                new Dictionary<string, string>( )
                {
                    {"One", "Sample one"},
                    {"Two", "Sample two"}
                };
            // Example3 Used for demonstrating how to build
            IEnumerable<IEnumerable<string>> Example4 =
                DicTest3.Select(a =>
                    Enumerable.Repeat(a.Value, 1)
                    .Concat(Enumerable.Repeat(a.Value, 1)));
            Debug.WriteLine(string.Format("{0} Example3",
                MethodInfo.GetCurrentMethod( ).Name));
            CheckCastAndResult(Example3);

        }

        /// <summary>
        /// A quick little test that the results of the cast has yielded
        /// an IEnumerable&lt;IEnumerable&lt;&gt;&gt;.
        /// </summary>
        /// <param name="checkObject">
        /// The object which is being checked
        /// </param>
        /// <remarks>
        /// Could be expended, quite simple at present.
        /// </remarks>
        internal void CheckCastAndResult<TSource>(IEnumerable<IEnumerable<TSource>> checkObject)
        {

            Debug.WriteLine(string.Format("{0}", checkObject.GetType( ).ToString( )));
        }
    }
}
```

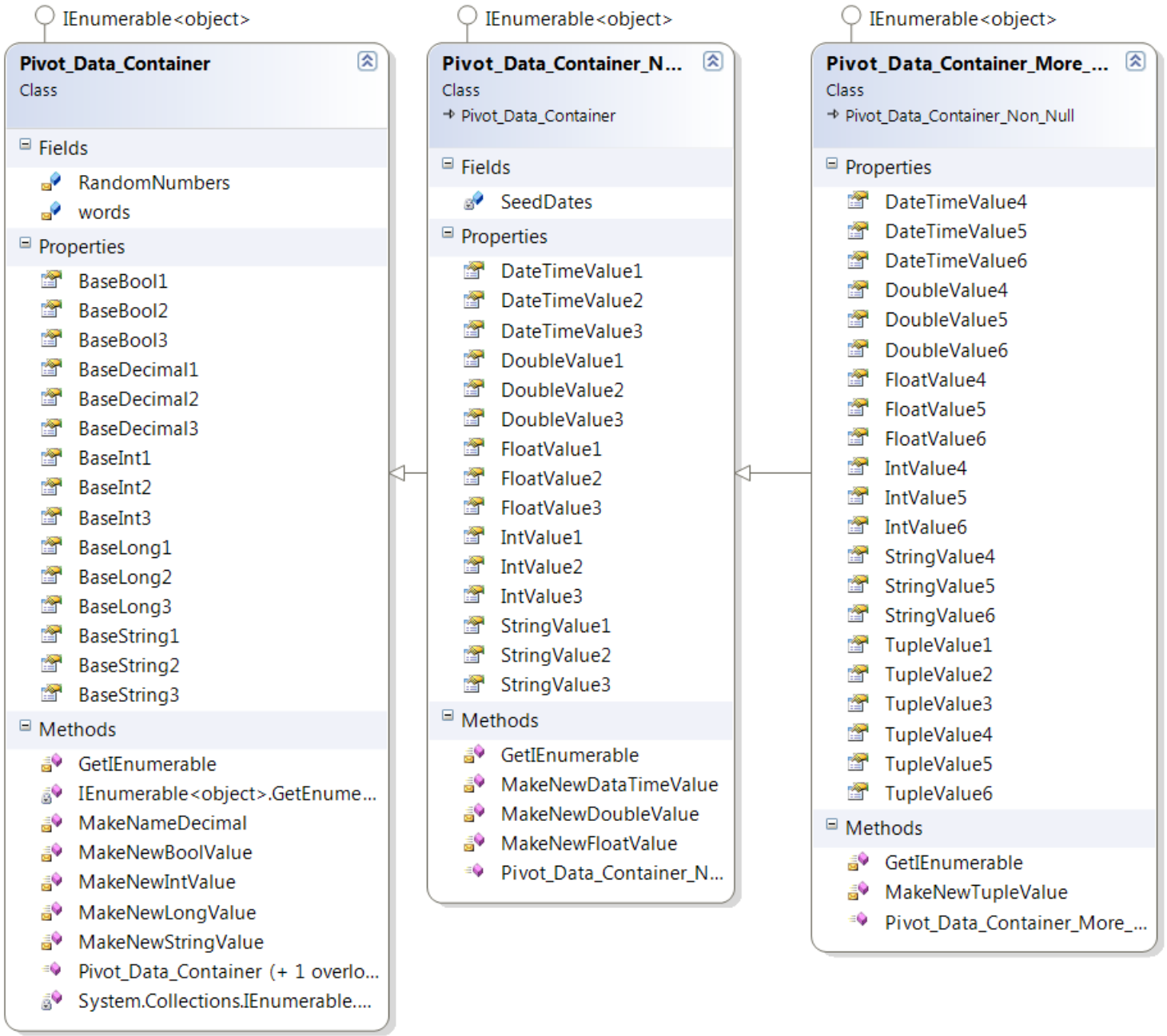
Classes that Implement IEnumerable<Of Source Type> to Enable Building IEnumerable<IEnumerable<Of Source Type>>

Example Use

The following is a simple code fragment that demonstrates the use of the IEnumerable interface on the classes included below.

```
IEnumerable<IEnumerable<object>> RaggedVeryShortStart =
    new IEnumerable<Object>[]
{
    new Pivot_Data_Container(),
    new Pivot_Data_Container_Non_Null(),
    new Pivot_Data_Container_More_Fields(),
};
```

Class Diagram



Pivot Data Container Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LINQ_Extensions
{
    internal class Pivot_Data_Container:IEnumerable<object>
    {
        internal static Random RandomNumbers = null;
        internal List<string> words = new List<string>( )
        {
"In ","data ","processing, ","a ","pivot ","table ","is ","a ","data ","summarization ",
"tool ","found ","in ","data ","visualization ","programs ","such ","as ","spreadsheets ",
"or ","business ","intelligence ","software. ","Among ","other ","functions, ",
"pivot-table ","tools ","can ","automatically ","sort, ","count, ","total ","or ",
"give ","the ","average ","of ","the ","data ","stored ","in ","one ","table ","or ",
"spreadsheet. ","It ","displays ","the ","results ","in ","a ","second ","table ",
"(called ","a ","pivot ","table) ","showing ","the ","summarized ","data. ",
"Pivot ","tables ","are ","also ","useful ","for ","quickly ","creating ",
"unweighted ","cross ","tabulations. ","The ","user ","sets ","up ","and ",
"changes ","the ","summary's ","structure ","by ","dragging ","and ","dropping ",
"fields ","graphically. ","This ","rotation ","or ","pivoting ","of ","the ",
"summary ","table ","gives ","the ","concept ","its ","name. ","The ","term ",
"pivot ","table ","is ","a ","generic ","phrase ","used ","by ","multiple ","vendors. ",
"However, ","Microsoft ","Corporation ","has ","trademarked ","the ","specific ","form ",
"PivotTable.[1][2] "," "," "," "," "," "," "," "," "," "," "," "," "," ",""
        };
        public int BaseInt1
        {
            get;
            set;
        }
        public int BaseInt2
        {
            get;
            set;
        }
        public int BaseInt3
        {
            get;
            set;
        }
```

```
public long BaseLong1
{
    get;
    set;
}
public long BaseLong2
{
    get;
    set;
}
public long BaseLong3
{
    get;
    set;
}
public decimal BaseDecimal1
{
    get;
    set;
}
public decimal BaseDecimal2
{
    get;
    set;
}
public decimal BaseDecimal3
{
    get;
    set;
}
public string BaseString1
{
    get;
    set;
}
public string BaseString2
{
    get;
    set;
}
public string BaseString3
{
    get;
    set;
```

```
}
public bool BaseBool1
{
    get;
    set;
}
public bool BaseBool2
{
    get;
    set;
}
public bool BaseBool3
{
    get;
    set;
}
public Pivot_Data_Container( )
{
    if (Object.ReferenceEquals(Pivot_Data_Container.RandomNumbers, null))
        RandomNumbers = new Random( );
    this.BaseBool1 = MakeNewBoolValue( );
    this.BaseBool2 = MakeNewBoolValue( );
    this.BaseBool3 = MakeNewBoolValue( );
    this.BaseInt1 = MakeNewIntValue( );
    this.BaseInt2 = MakeNewIntValue( );
    this.BaseInt3 = MakeNewIntValue( );
    this.BaseLong1 = MakeNewLongValue( );
    this.BaseDecimal1 = MakeNameDecimal( );
    this.BaseDecimal2 = MakeNameDecimal( );
    this.BaseDecimal3 = MakeNameDecimal( );
    this.BaseString1 = MakeNewStringValue( );
    this.BaseString1 = MakeNewStringValue( );
    this.BaseString1 = MakeNewStringValue( );
}

public Pivot_Data_Container(int Int1, int Int2, int Int3,
    long Long1, long Long2, long Long3,
    decimal Decimal1, decimal Decimal2, decimal Decimal3,
    string String1, string String2, string String3,
    bool Bool1, bool Bool2, bool Bool3)
{
    this.BaseInt1 = Int1;
    this.BaseInt2 = Int2;
    this.BaseInt3 = Int3;
```

```
        this.BaseLong1 = Long1;
        this.BaseLong2 = Long2;
        this.BaseLong3 = Long1;
        this.BaseDecimal1 = Decimal1;
        this.BaseString1 = String1;
        this.BaseString2 = String2;
        this.BaseString3 = String3;
        this.BaseBool1 = BaseBool1;
        this.BaseBool2 = BaseBool2;
        this.BaseBool3 = BaseBool3;
    }

    internal virtual IEnumerable<object> GetIEnumerable( )
    {
        return new object[]
        {
            this.BaseBool1, this.BaseBool2, this.BaseBool3,
            this.BaseDecimal1, this.BaseDecimal2, this.BaseDecimal3,
            this.BaseInt1, this.BaseInt2, this.BaseInt3,
            this.BaseLong1, this.BaseLong2, this.BaseLong3,
            this.BaseString1, this.BaseString2, this.BaseString3,
        };
    }

    internal int MakeNewIntValue( )
    {
        return RandomNumbers.Next(Int32.MinValue + 1, Int32.MaxValue - 1);
    }

    internal bool MakeNewBoolValue( )
    {
        int flip = RandomNumbers.Next(0, 2);
        bool res = flip == 0;
        return res;
        //return RandomNumbers.Next(0, 1) == 1 ? true : false;
    }
    internal long MakeNewLongValue( )
    {
        return MakeNewIntValue( ) + MakeNewIntValue( );
    }
    internal decimal MakeNameDecimal( )
    {
        decimal retVal = (decimal) (RandomNumbers.NextDouble( ) *
            (Math.Pow(10, RandomNumbers.Next(-20, 20))));
```

```
            return retVal;
        }
        internal string MakeNewStringValue( )
        {
            string result = "Random Words. ";
            var wordPicks = (from num in Enumerable.Range(0, 20)
                             let index = RandomNumbers.Next(0, words.Count)
                             select words.ElementAt(index))
                            .Aggregate(new StringBuilder(result),
                            (builder, word) => builder.Append(word));
                            //(word, builder) =>
                            //    {
                            //        builder.App
                            //    }
            return wordPicks.ToString( );
        }


        #region IEnumerable<object> Members

        IEnumerator<object> IEnumerable<object>.GetEnumerator( )
        {
            return this.GetIEnumerable( ).GetEnumerator();
        }

        #endregion

        #region IEnumerable Members

        System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator( )
        {
            return this.GetIEnumerable( ).GetEnumerator( );
        }

        #endregion
    }
}
```

Pivot Data Container Non Null Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using LINQ_Extension_Methods;

namespace LINQ_Extensions
{
    internal class Pivot_Data_Container_Non_Null
        : Pivot_Data_Container
        , IEnumerable<Object>
    {
        private List<DateTime> SeedDates = new List<DateTime>( )
        {
            new DateTime(2001, 1, 1, 12, 0, 0), new DateTime(2012, 5, 19, 0, 0, 0),
            new DateTime(2012, 3, 28, 0, 0, 0), new DateTime(1962, 1, 1, 12, 0, 0),
            new DateTime(1788, 5, 19, 0, 0, 0), new DateTime(1901, 3, 28, 0, 0, 0),
            new DateTime(2001, 1, 1, 12, 0, 0), new DateTime(2012, 5, 19, 0, 0, 0),
            new DateTime(2012, 3, 28, 0, 0, 0), new DateTime(2101, 2, 2, 12, 0, 0),
            new DateTime(2013, 4, 10, 0, 0, 0), new DateTime(2013, 9, 12, 0, 0, 0),
            new DateTime(1972, 1, 1, 12, 0, 0), new DateTime(1962, 11, 12, 0, 0, 0),
            new DateTime(1801, 3, 1, 0, 0, 0),  new DateTime(1972, 1, 1, 12, 0, 0),
            new DateTime(1962, 11, 12, 0, 0, 0), new DateTime(1801, 3, 1, 0, 0, 0),
            new DateTime(1968, 1, 4, 12, 0, 0), new DateTime(2020, 4, 13, 0, 0, 0),
            new DateTime(1799, 1, 1, 0, 0, 0)
        };

        public int IntValue1
        {
            get;
            set;
        }
        public int IntValue2
        {
            get;
            set;
        }
        public int IntValue3
        {
            get;
            set;
        }
        public float FloatValue1
```

```
{
    get;
    set;
}
public float FloatValue2
{
    get;
    set;
}
public float FloatValue3
{
    get;
    set;
}
public double DoubleValue1
{
    get;
    set;
}
public double DoubleValue2
{
    get;
    set;
}
public double DoubleValue3
{
    get;
    set;
}
public string StringValue1
{
    get;
    set;
}
public string StringValue2
{
    get;
    set;
}
public string StringValue3
{
    get;
    set;
}
```

```
public DateTime DateTimeValue1
{
    get;
    set;
}
public DateTime DateTimeValue2
{
    get;
    set;
}
public DateTime DateTimeValue3
{
    get;
    set;
}
public Pivot_Data_Container_Non_Null( )
    : base( )
{
    this.IntValue1 = MakeNewIntValue( );
    this.IntValue2 = MakeNewIntValue( );
    this.IntValue3 = MakeNewIntValue( );
    this.FloatValue1 = MakeNewFloatValue( );
    this.FloatValue2 = MakeNewFloatValue( );
    this.FloatValue3 = MakeNewFloatValue( );
    this.DoubleValue1 = MakeNewDoubleValue( );
    this.DoubleValue2 = MakeNewDoubleValue( );
    this.DoubleValue3 = MakeNewDoubleValue( );
    this.StringValue1 = MakeNewStringValue( );
    this.StringValue2 = MakeNewStringValue( );
    this.StringValue3 = MakeNewStringValue( );
    this.DateTimeValue1 = MakeNewDataTimeValue( );
    this.DateTimeValue2 = MakeNewDataTimeValue( );
    this.DateTimeValue3 = MakeNewDataTimeValue( );
}
public Pivot_Data_Container_Non_Null(
    bool Bool1, bool Bool2, bool Bool3,
    int Int1, int Int2, int Int3,
    int Int4, int Int5, int Int6,
    long Long1, long Long2, long Long3,
    decimal Decimal1, decimal Decimal2, decimal Decimal3,
    float Float1, float Float2, float Float3,
    double Double1, double Double2, double Double3,
    string String1, string String2, string String3,
    string String4, string String5, string String6,
```

```
    DateTime DateTime1, DateTime DateTime2, DateTime DataTime3)
    //            : this()
    : base(Int1, Int2, Int3, Long1, Long2, Long3, Decimal1, Decimal2, Decimal3,
    String1, String2, String3, Bool1, Bool2, Bool3)
{
    this.IntValue1 = Int4;
    this.IntValue2 = Int5;
    this.IntValue3 = Int6;
    this.FloatValue1 = Float1;
    this.FloatValue2 = Float2;
    this.FloatValue3 = Float3;
    this.DoubleValue1 = Double1;
    this.DoubleValue2 = Double2;
    this.DoubleValue3 = Double3;
    this.StringValue1 = String4;
    this.StringValue2 = String5;
    this.StringValue3 = String6;
    this.DateTimeValue1 = DateTime1;
    this.DateTimeValue2 = DateTime2;
    this.DateTimeValue3 = DataTime3;
}
internal override IEnumerable<Object> GetIEnumerable( )
{
    IEnumerable<Object> baseValues = base.GetIEnumerable( );
    IEnumerable<object> thisValues = new object[]
        {
            this.IntValue1, this.IntValue2, this.IntValue3,
            this.FloatValue1, this.FloatValue2, this.FloatValue3,
            this.DoubleValue1, this.DoubleValue2, this.DoubleValue3,
            this.StringValue1, this.StringValue2, this.StringValue3,
            this.DateTimeValue1, this.DateTimeValue2, this.DateTimeValue3
        };
    return baseValues.Concat(thisValues);
}

internal float MakeNewFloatValue( )
{
    float retVal = (float) (RandomNumbers.NextDouble( ) *
        (Math.Pow(10, RandomNumbers.Next(-30, 30))));
    return retVal;
}
internal double MakeNewDoubleValue( )
{
    return (RandomNumbers.NextDouble( ) *
```

```
                (Math.Pow(10, RandomNumbers.Next(-100, 100))));
        }
        internal DateTime MakeNewDataTimeValue()
        {
            return SeedDates
                .ElementAt(RandomNumbers.Next(0, SeedDates.Count))
                .AddDays(RandomNumbers.Next(-20000, 20000));
        }

    }
}
```

Pivot Data Container More Fields

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using LINQ_Extension_Methods;

namespace LINQ_Extensions
{
    internal class Pivot_Data_Container_More_Fields
        : Pivot_Data_Container_Non_Null
        , IEnumerable<Object>
    {

        public int IntValue4
        {
            get;
            set;
        }
        public int IntValue5
        {
            get;
            set;
        }
        public int IntValue6
        {
            get;
            set;
        }
        public float FloatValue4
        {
```

```
    get;
    set;
}
public float FloatValue5
{
    get;
    set;
}
public float FloatValue6
{
    get;
    set;
}
public double DoubleValue4
{
    get;
    set;
}
public double DoubleValue5
{
    get;
    set;
}
public double DoubleValue6
{
    get;
    set;
}
public string StringValue4
{
    get;
    set;
}
public string StringValue5
{
    get;
    set;
}
public string StringValue6
{
    get;
    set;
}
public DateTime DateTimeValue4
```

```
{
    get;
    set;
}
public DateTime DateTimeValue5
{
    get;
    set;
}
public DateTime DateTimeValue6
{
    get;
    set;
}
public Tuple<int, string> TupleValue1
{
    get;
    set;
}
public Tuple<int, string> TupleValue2
{
    get;
    set;
}
public Tuple<int, string> TupleValue3
{
    get;
    set;
}
public Tuple<int, string> TupleValue4
{
    get;
    set;
}
public Tuple<int, string> TupleValue5
{
    get;
    set;
}
public Tuple<int, string> TupleValue6
{
    get;
    set;
}
```

```
public Pivot_Data_Container_More_Fields( )
    : base( )
{
    this.IntValue4 = MakeNewIntValue( );
    this.IntValue5 = MakeNewIntValue( );
    this.IntValue6 = MakeNewIntValue( );
    this.FloatValue4 = MakeNewFloatValue( );
    this.FloatValue5 = MakeNewFloatValue( );
    this.FloatValue6 = MakeNewFloatValue( );
    this.DoubleValue4 = MakeNewDoubleValue( );
    this.DoubleValue5 = MakeNewDoubleValue( );
    this.DoubleValue6 = MakeNewDoubleValue( );
    this.StringValue4 = MakeNewStringValue( );
    this.StringValue5 = MakeNewStringValue( );
    this.StringValue6 = MakeNewStringValue( );
    this.DateTimeValue4 = MakeNewDataTimeValue( );
    this.DateTimeValue5 = MakeNewDataTimeValue( );
    this.DateTimeValue6 = MakeNewDataTimeValue( );
    this.TupleValue1 = MakeNewTupleValue( );
    this.TupleValue2 = MakeNewTupleValue( );
    this.TupleValue3 = MakeNewTupleValue( );
    this.TupleValue4 = MakeNewTupleValue( );
    this.TupleValue5 = MakeNewTupleValue( );
    this.TupleValue6 = MakeNewTupleValue( );
}
public Pivot_Data_Container_More_Fields(
    bool Bool1, bool Bool2, bool Bool3,
    int Int1, int Int2, int Int3,
    int Int4, int Int5, int Int6,
    int Int7, int Int8, int Int9,
    long Long1, long Long2, long Long3,
    decimal Decimal1, decimal Decimal2, decimal Decimal3,
    float Float1, float Float2, float Float3,
    float Float4, float Fload5, float Float6,
    double Double1, double Double2, double Double3,
    double Double4, double Double5, double Double6,
    string String1, string String2, string String3,
    string String4, string String5, string String6,
    string String7, string String8, string String9,
    DateTime DateTime1, DateTime DateTime2, DateTime DataTime3,
    DateTime DataTime4, DateTime DataTime5, DateTime DateTime6,
    Tuple<int, string> Tuple1, Tuple<int, string> Tuple2, Tuple<int, string> Tuple3,
    Tuple<int, string> Tuple4, Tuple<int, string> Tuple5, Tuple<int, string> Tuple6)
    : base(Bool1, Bool2, Bool3, Int1, Int2, Int3, Int4, Int5, Int6,
```

```
    Long1, Long2, Long3, Decimal1, Decimal2, Decimal3, Float1, Float2, Float3,
    Double1, Double2, Double3, String1, String2, String3, String4, String5, String6,
    DateTime1, DateTime2, DataTime3)
{
    this.IntValue4 = Int7;
    this.IntValue5 = Int8;
    this.IntValue6 = Int9;
    this.FloatValue4 = Float4;
    this.FloatValue5 = Fload5;
    this.FloatValue6 = Float6;
    this.DoubleValue4 = Double4;
    this.DoubleValue5 = Double5;
    this.DoubleValue6 = Double6;
    this.StringValue4 = String7;
    this.StringValue5 = String8;
    this.StringValue6 = String9;
    this.DateTimeValue4 = DataTime4;
    this.DateTimeValue5 = DataTime5;
    this.DateTimeValue6 = DateTime6;
    if (Object.ReferenceEquals(Tuple1, null))
        this.TupleValue1 = null;
    else
        this.TupleValue1 = Tuple.Create<int, string>(Tuple1.Item1, Tuple1.Item2);
    if (Object.ReferenceEquals(Tuple2, null))
        this.TupleValue2 = null;
    else
        this.TupleValue2 = Tuple.Create<int, string>(Tuple2.Item1, Tuple2.Item2);

    if (Object.ReferenceEquals(Tuple3, null))
        this.TupleValue3 = null;
    else
        this.TupleValue3 = Tuple.Create<int, string>(Tuple3.Item1, Tuple3.Item2);
    if (Object.ReferenceEquals(Tuple4, null))
        this.TupleValue4 = null;
    else
        this.TupleValue4 = Tuple.Create<int, string>(Tuple4.Item1, Tuple4.Item2);
    if (Object.ReferenceEquals(Tuple5, null))
        this.TupleValue5 = null;
    else
        this.TupleValue5 = Tuple.Create<int, string>(Tuple5.Item1, Tuple5.Item2);
    if (Object.ReferenceEquals(Tuple6, null))
        this.TupleValue6 = null;
    else
        this.TupleValue6 = Tuple.Create<int, string>(Tuple6.Item1, Tuple6.Item2);
```

```
        }
        internal override IEnumerable<Object> GetIEnumerable( )
        {
            IEnumerable<Object> baseValues = base.GetIEnumerable( );
            IEnumerable<object> thisValues = new object[]
                {
                    this.IntValue4, this.IntValue5, this.IntValue6,
                    this.FloatValue4, this.FloatValue5, this.FloatValue6,
                    this.DoubleValue4, this.DoubleValue5, this.DoubleValue6,
                    this.StringValue4, this.StringValue5, this.StringValue6,
                    this.DateTimeValue4, this.DateTimeValue5, this.DateTimeValue6,
                    this.TupleValue1, this.TupleValue2, this.TupleValue3,
                    this.TupleValue4, this.TupleValue5, this.TupleValue6
                };
            return baseValues.Concat(thisValues);
        }

        internal Tuple<int, string> MakeNewTupleValue()
        {
            if (RandomNumbers.Next(0, 30) == 0)
                return null;
            else
                return Tuple.Create<int, string>(
                    MakeNewIntValue(),
                    MakeNewStringValue());
        }

    }
}
```