﻿'------------------------------------------------------------------------------

' <auto-generated>

' This code was generated by a tool.

' Runtime Version:4.0.30319.42000

'

' Changes to this file may cause incorrect behavior and will be lost if

' the code is regenerated.

' </auto-generated>

'------------------------------------------------------------------------------

Option Strict Off

Option Explicit On

'''<summary>

'''Represents a strongly typed in-memory cache of data.

'''</summary>

<Global.System.Serializable(), \_

Global.System.ComponentModel.DesignerCategoryAttribute("code"), \_

Global.System.ComponentModel.ToolboxItem(true), \_

Global.System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedDataSetSchema"), \_

Global.System.Xml.Serialization.XmlRootAttribute("InvoiceDataSet"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.DataSet")> \_

Partial Public Class InvoiceDataSet

Inherits Global.System.Data.DataSet

Private tableadmin As adminDataTable

Private tablestock As stockDataTable

Private \_schemaSerializationMode As Global.System.Data.SchemaSerializationMode = Global.System.Data.SchemaSerializationMode.IncludeSchema

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub New()

MyBase.New

Me.BeginInit

Me.InitClass

Dim schemaChangedHandler As Global.System.ComponentModel.CollectionChangeEventHandler = AddressOf Me.SchemaChanged

AddHandler MyBase.Tables.CollectionChanged, schemaChangedHandler

AddHandler MyBase.Relations.CollectionChanged, schemaChangedHandler

Me.EndInit

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Sub New(ByVal info As Global.System.Runtime.Serialization.SerializationInfo, ByVal context As Global.System.Runtime.Serialization.StreamingContext)

MyBase.New(info, context, false)

If (Me.IsBinarySerialized(info, context) = true) Then

Me.InitVars(false)

Dim schemaChangedHandler1 As Global.System.ComponentModel.CollectionChangeEventHandler = AddressOf Me.SchemaChanged

AddHandler Me.Tables.CollectionChanged, schemaChangedHandler1

AddHandler Me.Relations.CollectionChanged, schemaChangedHandler1

Return

End If

Dim strSchema As String = CType(info.GetValue("XmlSchema", GetType(String)),String)

If (Me.DetermineSchemaSerializationMode(info, context) = Global.System.Data.SchemaSerializationMode.IncludeSchema) Then

Dim ds As Global.System.Data.DataSet = New Global.System.Data.DataSet()

ds.ReadXmlSchema(New Global.System.Xml.XmlTextReader(New Global.System.IO.StringReader(strSchema)))

If (Not (ds.Tables("admin")) Is Nothing) Then

MyBase.Tables.Add(New adminDataTable(ds.Tables("admin")))

End If

If (Not (ds.Tables("stock")) Is Nothing) Then

MyBase.Tables.Add(New stockDataTable(ds.Tables("stock")))

End If

Me.DataSetName = ds.DataSetName

Me.Prefix = ds.Prefix

Me.Namespace = ds.Namespace

Me.Locale = ds.Locale

Me.CaseSensitive = ds.CaseSensitive

Me.EnforceConstraints = ds.EnforceConstraints

Me.Merge(ds, false, Global.System.Data.MissingSchemaAction.Add)

Me.InitVars

Else

Me.ReadXmlSchema(New Global.System.Xml.XmlTextReader(New Global.System.IO.StringReader(strSchema)))

End If

Me.GetSerializationData(info, context)

Dim schemaChangedHandler As Global.System.ComponentModel.CollectionChangeEventHandler = AddressOf Me.SchemaChanged

AddHandler MyBase.Tables.CollectionChanged, schemaChangedHandler

AddHandler Me.Relations.CollectionChanged, schemaChangedHandler

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Browsable(false), \_

Global.System.ComponentModel.DesignerSerializationVisibility(Global.System.ComponentModel.DesignerSerializationVisibility.Content)> \_

Public ReadOnly Property admin() As adminDataTable

Get

Return Me.tableadmin

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Browsable(false), \_

Global.System.ComponentModel.DesignerSerializationVisibility(Global.System.ComponentModel.DesignerSerializationVisibility.Content)> \_

Public ReadOnly Property stock() As stockDataTable

Get

Return Me.tablestock

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.BrowsableAttribute(true), \_

Global.System.ComponentModel.DesignerSerializationVisibilityAttribute(Global.System.ComponentModel.DesignerSerializationVisibility.Visible)> \_

Public Overrides Property SchemaSerializationMode() As Global.System.Data.SchemaSerializationMode

Get

Return Me.\_schemaSerializationMode

End Get

Set

Me.\_schemaSerializationMode = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.DesignerSerializationVisibilityAttribute(Global.System.ComponentModel.DesignerSerializationVisibility.Hidden)> \_

Public Shadows ReadOnly Property Tables() As Global.System.Data.DataTableCollection

Get

Return MyBase.Tables

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.DesignerSerializationVisibilityAttribute(Global.System.ComponentModel.DesignerSerializationVisibility.Hidden)> \_

Public Shadows ReadOnly Property Relations() As Global.System.Data.DataRelationCollection

Get

Return MyBase.Relations

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Sub InitializeDerivedDataSet()

Me.BeginInit

Me.InitClass

Me.EndInit

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Overrides Function Clone() As Global.System.Data.DataSet

Dim cln As InvoiceDataSet = CType(MyBase.Clone,InvoiceDataSet)

cln.InitVars

cln.SchemaSerializationMode = Me.SchemaSerializationMode

Return cln

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Function ShouldSerializeTables() As Boolean

Return false

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Function ShouldSerializeRelations() As Boolean

Return false

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Sub ReadXmlSerializable(ByVal reader As Global.System.Xml.XmlReader)

If (Me.DetermineSchemaSerializationMode(reader) = Global.System.Data.SchemaSerializationMode.IncludeSchema) Then

Me.Reset

Dim ds As Global.System.Data.DataSet = New Global.System.Data.DataSet()

ds.ReadXml(reader)

If (Not (ds.Tables("admin")) Is Nothing) Then

MyBase.Tables.Add(New adminDataTable(ds.Tables("admin")))

End If

If (Not (ds.Tables("stock")) Is Nothing) Then

MyBase.Tables.Add(New stockDataTable(ds.Tables("stock")))

End If

Me.DataSetName = ds.DataSetName

Me.Prefix = ds.Prefix

Me.Namespace = ds.Namespace

Me.Locale = ds.Locale

Me.CaseSensitive = ds.CaseSensitive

Me.EnforceConstraints = ds.EnforceConstraints

Me.Merge(ds, false, Global.System.Data.MissingSchemaAction.Add)

Me.InitVars

Else

Me.ReadXml(reader)

Me.InitVars

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Function GetSchemaSerializable() As Global.System.Xml.Schema.XmlSchema

Dim stream As Global.System.IO.MemoryStream = New Global.System.IO.MemoryStream()

Me.WriteXmlSchema(New Global.System.Xml.XmlTextWriter(stream, Nothing))

stream.Position = 0

Return Global.System.Xml.Schema.XmlSchema.Read(New Global.System.Xml.XmlTextReader(stream), Nothing)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Overloads Sub InitVars()

Me.InitVars(true)

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Overloads Sub InitVars(ByVal initTable As Boolean)

Me.tableadmin = CType(MyBase.Tables("admin"),adminDataTable)

If (initTable = true) Then

If (Not (Me.tableadmin) Is Nothing) Then

Me.tableadmin.InitVars

End If

End If

Me.tablestock = CType(MyBase.Tables("stock"),stockDataTable)

If (initTable = true) Then

If (Not (Me.tablestock) Is Nothing) Then

Me.tablestock.InitVars

End If

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Sub InitClass()

Me.DataSetName = "InvoiceDataSet"

Me.Prefix = ""

Me.Namespace = "http://tempuri.org/InvoiceDataSet.xsd"

Me.EnforceConstraints = true

Me.SchemaSerializationMode = Global.System.Data.SchemaSerializationMode.IncludeSchema

Me.tableadmin = New adminDataTable()

MyBase.Tables.Add(Me.tableadmin)

Me.tablestock = New stockDataTable()

MyBase.Tables.Add(Me.tablestock)

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Function ShouldSerializeadmin() As Boolean

Return false

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Function ShouldSerializestock() As Boolean

Return false

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Sub SchemaChanged(ByVal sender As Object, ByVal e As Global.System.ComponentModel.CollectionChangeEventArgs)

If (e.Action = Global.System.ComponentModel.CollectionChangeAction.Remove) Then

Me.InitVars

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Shared Function GetTypedDataSetSchema(ByVal xs As Global.System.Xml.Schema.XmlSchemaSet) As Global.System.Xml.Schema.XmlSchemaComplexType

Dim ds As InvoiceDataSet = New InvoiceDataSet()

Dim type As Global.System.Xml.Schema.XmlSchemaComplexType = New Global.System.Xml.Schema.XmlSchemaComplexType()

Dim sequence As Global.System.Xml.Schema.XmlSchemaSequence = New Global.System.Xml.Schema.XmlSchemaSequence()

Dim any As Global.System.Xml.Schema.XmlSchemaAny = New Global.System.Xml.Schema.XmlSchemaAny()

any.Namespace = ds.Namespace

sequence.Items.Add(any)

type.Particle = sequence

Dim dsSchema As Global.System.Xml.Schema.XmlSchema = ds.GetSchemaSerializable

If xs.Contains(dsSchema.TargetNamespace) Then

Dim s1 As Global.System.IO.MemoryStream = New Global.System.IO.MemoryStream()

Dim s2 As Global.System.IO.MemoryStream = New Global.System.IO.MemoryStream()

Try

Dim schema As Global.System.Xml.Schema.XmlSchema = Nothing

dsSchema.Write(s1)

Dim schemas As Global.System.Collections.IEnumerator = xs.Schemas(dsSchema.TargetNamespace).GetEnumerator

Do While schemas.MoveNext

schema = CType(schemas.Current,Global.System.Xml.Schema.XmlSchema)

s2.SetLength(0)

schema.Write(s2)

If (s1.Length = s2.Length) Then

s1.Position = 0

s2.Position = 0

Do While ((s1.Position <> s1.Length) \_

AndAlso (s1.ReadByte = s2.ReadByte))

Loop

If (s1.Position = s1.Length) Then

Return type

End If

End If

Loop

Finally

If (Not (s1) Is Nothing) Then

s1.Close

End If

If (Not (s2) Is Nothing) Then

s2.Close

End If

End Try

End If

xs.Add(dsSchema)

Return type

End Function

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Delegate Sub adminRowChangeEventHandler(ByVal sender As Object, ByVal e As adminRowChangeEvent)

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Delegate Sub stockRowChangeEventHandler(ByVal sender As Object, ByVal e As stockRowChangeEvent)

'''<summary>

'''Represents the strongly named DataTable class.

'''</summary>

<Global.System.Serializable(), \_

Global.System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedTableSchema")> \_

Partial Public Class adminDataTable

Inherits Global.System.Data.TypedTableBase(Of adminRow)

Private columnID As Global.System.Data.DataColumn

Private columnusername As Global.System.Data.DataColumn

Private columnpassword As Global.System.Data.DataColumn

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub New()

MyBase.New

Me.TableName = "admin"

Me.BeginInit

Me.InitClass

Me.EndInit

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Sub New(ByVal table As Global.System.Data.DataTable)

MyBase.New

Me.TableName = table.TableName

If (table.CaseSensitive <> table.DataSet.CaseSensitive) Then

Me.CaseSensitive = table.CaseSensitive

End If

If (table.Locale.ToString <> table.DataSet.Locale.ToString) Then

Me.Locale = table.Locale

End If

If (table.Namespace <> table.DataSet.Namespace) Then

Me.Namespace = table.Namespace

End If

Me.Prefix = table.Prefix

Me.MinimumCapacity = table.MinimumCapacity

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Sub New(ByVal info As Global.System.Runtime.Serialization.SerializationInfo, ByVal context As Global.System.Runtime.Serialization.StreamingContext)

MyBase.New(info, context)

Me.InitVars

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public ReadOnly Property IDColumn() As Global.System.Data.DataColumn

Get

Return Me.columnID

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public ReadOnly Property usernameColumn() As Global.System.Data.DataColumn

Get

Return Me.columnusername

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public ReadOnly Property passwordColumn() As Global.System.Data.DataColumn

Get

Return Me.columnpassword

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Browsable(false)> \_

Public ReadOnly Property Count() As Integer

Get

Return Me.Rows.Count

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Default ReadOnly Property Item(ByVal index As Integer) As adminRow

Get

Return CType(Me.Rows(index),adminRow)

End Get

End Property

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Event adminRowChanging As adminRowChangeEventHandler

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Event adminRowChanged As adminRowChangeEventHandler

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Event adminRowDeleting As adminRowChangeEventHandler

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Event adminRowDeleted As adminRowChangeEventHandler

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Overloads Sub AddadminRow(ByVal row As adminRow)

Me.Rows.Add(row)

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Overloads Function AddadminRow(ByVal username As String, ByVal password As String) As adminRow

Dim rowadminRow As adminRow = CType(Me.NewRow,adminRow)

Dim columnValuesArray() As Object = New Object() {Nothing, username, password}

rowadminRow.ItemArray = columnValuesArray

Me.Rows.Add(rowadminRow)

Return rowadminRow

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Function FindByID(ByVal ID As Integer) As adminRow

Return CType(Me.Rows.Find(New Object() {ID}),adminRow)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Overrides Function Clone() As Global.System.Data.DataTable

Dim cln As adminDataTable = CType(MyBase.Clone,adminDataTable)

cln.InitVars

Return cln

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Function CreateInstance() As Global.System.Data.DataTable

Return New adminDataTable()

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Sub InitVars()

Me.columnID = MyBase.Columns("ID")

Me.columnusername = MyBase.Columns("username")

Me.columnpassword = MyBase.Columns("password")

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Sub InitClass()

Me.columnID = New Global.System.Data.DataColumn("ID", GetType(Integer), Nothing, Global.System.Data.MappingType.Element)

MyBase.Columns.Add(Me.columnID)

Me.columnusername = New Global.System.Data.DataColumn("username", GetType(String), Nothing, Global.System.Data.MappingType.Element)

MyBase.Columns.Add(Me.columnusername)

Me.columnpassword = New Global.System.Data.DataColumn("password", GetType(String), Nothing, Global.System.Data.MappingType.Element)

MyBase.Columns.Add(Me.columnpassword)

Me.Constraints.Add(New Global.System.Data.UniqueConstraint("Constraint1", New Global.System.Data.DataColumn() {Me.columnID}, true))

Me.columnID.AutoIncrement = true

Me.columnID.AutoIncrementSeed = -1

Me.columnID.AutoIncrementStep = -1

Me.columnID.AllowDBNull = false

Me.columnID.Unique = true

Me.columnusername.MaxLength = 255

Me.columnpassword.MaxLength = 255

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Function NewadminRow() As adminRow

Return CType(Me.NewRow,adminRow)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Function NewRowFromBuilder(ByVal builder As Global.System.Data.DataRowBuilder) As Global.System.Data.DataRow

Return New adminRow(builder)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Function GetRowType() As Global.System.Type

Return GetType(adminRow)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Sub OnRowChanged(ByVal e As Global.System.Data.DataRowChangeEventArgs)

MyBase.OnRowChanged(e)

If (Not (Me.adminRowChangedEvent) Is Nothing) Then

RaiseEvent adminRowChanged(Me, New adminRowChangeEvent(CType(e.Row,adminRow), e.Action))

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Sub OnRowChanging(ByVal e As Global.System.Data.DataRowChangeEventArgs)

MyBase.OnRowChanging(e)

If (Not (Me.adminRowChangingEvent) Is Nothing) Then

RaiseEvent adminRowChanging(Me, New adminRowChangeEvent(CType(e.Row,adminRow), e.Action))

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Sub OnRowDeleted(ByVal e As Global.System.Data.DataRowChangeEventArgs)

MyBase.OnRowDeleted(e)

If (Not (Me.adminRowDeletedEvent) Is Nothing) Then

RaiseEvent adminRowDeleted(Me, New adminRowChangeEvent(CType(e.Row,adminRow), e.Action))

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Sub OnRowDeleting(ByVal e As Global.System.Data.DataRowChangeEventArgs)

MyBase.OnRowDeleting(e)

If (Not (Me.adminRowDeletingEvent) Is Nothing) Then

RaiseEvent adminRowDeleting(Me, New adminRowChangeEvent(CType(e.Row,adminRow), e.Action))

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub RemoveadminRow(ByVal row As adminRow)

Me.Rows.Remove(row)

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Shared Function GetTypedTableSchema(ByVal xs As Global.System.Xml.Schema.XmlSchemaSet) As Global.System.Xml.Schema.XmlSchemaComplexType

Dim type As Global.System.Xml.Schema.XmlSchemaComplexType = New Global.System.Xml.Schema.XmlSchemaComplexType()

Dim sequence As Global.System.Xml.Schema.XmlSchemaSequence = New Global.System.Xml.Schema.XmlSchemaSequence()

Dim ds As InvoiceDataSet = New InvoiceDataSet()

Dim any1 As Global.System.Xml.Schema.XmlSchemaAny = New Global.System.Xml.Schema.XmlSchemaAny()

any1.Namespace = "http://www.w3.org/2001/XMLSchema"

any1.MinOccurs = New Decimal(0)

any1.MaxOccurs = Decimal.MaxValue

any1.ProcessContents = Global.System.Xml.Schema.XmlSchemaContentProcessing.Lax

sequence.Items.Add(any1)

Dim any2 As Global.System.Xml.Schema.XmlSchemaAny = New Global.System.Xml.Schema.XmlSchemaAny()

any2.Namespace = "urn:schemas-microsoft-com:xml-diffgram-v1"

any2.MinOccurs = New Decimal(1)

any2.ProcessContents = Global.System.Xml.Schema.XmlSchemaContentProcessing.Lax

sequence.Items.Add(any2)

Dim attribute1 As Global.System.Xml.Schema.XmlSchemaAttribute = New Global.System.Xml.Schema.XmlSchemaAttribute()

attribute1.Name = "namespace"

attribute1.FixedValue = ds.Namespace

type.Attributes.Add(attribute1)

Dim attribute2 As Global.System.Xml.Schema.XmlSchemaAttribute = New Global.System.Xml.Schema.XmlSchemaAttribute()

attribute2.Name = "tableTypeName"

attribute2.FixedValue = "adminDataTable"

type.Attributes.Add(attribute2)

type.Particle = sequence

Dim dsSchema As Global.System.Xml.Schema.XmlSchema = ds.GetSchemaSerializable

If xs.Contains(dsSchema.TargetNamespace) Then

Dim s1 As Global.System.IO.MemoryStream = New Global.System.IO.MemoryStream()

Dim s2 As Global.System.IO.MemoryStream = New Global.System.IO.MemoryStream()

Try

Dim schema As Global.System.Xml.Schema.XmlSchema = Nothing

dsSchema.Write(s1)

Dim schemas As Global.System.Collections.IEnumerator = xs.Schemas(dsSchema.TargetNamespace).GetEnumerator

Do While schemas.MoveNext

schema = CType(schemas.Current,Global.System.Xml.Schema.XmlSchema)

s2.SetLength(0)

schema.Write(s2)

If (s1.Length = s2.Length) Then

s1.Position = 0

s2.Position = 0

Do While ((s1.Position <> s1.Length) \_

AndAlso (s1.ReadByte = s2.ReadByte))

Loop

If (s1.Position = s1.Length) Then

Return type

End If

End If

Loop

Finally

If (Not (s1) Is Nothing) Then

s1.Close

End If

If (Not (s2) Is Nothing) Then

s2.Close

End If

End Try

End If

xs.Add(dsSchema)

Return type

End Function

End Class

'''<summary>

'''Represents the strongly named DataTable class.

'''</summary>

<Global.System.Serializable(), \_

Global.System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedTableSchema")> \_

Partial Public Class stockDataTable

Inherits Global.System.Data.TypedTableBase(Of stockRow)

Private columnID As Global.System.Data.DataColumn

Private columnitems As Global.System.Data.DataColumn

Private columnquantity As Global.System.Data.DataColumn

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub New()

MyBase.New

Me.TableName = "stock"

Me.BeginInit

Me.InitClass

Me.EndInit

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Sub New(ByVal table As Global.System.Data.DataTable)

MyBase.New

Me.TableName = table.TableName

If (table.CaseSensitive <> table.DataSet.CaseSensitive) Then

Me.CaseSensitive = table.CaseSensitive

End If

If (table.Locale.ToString <> table.DataSet.Locale.ToString) Then

Me.Locale = table.Locale

End If

If (table.Namespace <> table.DataSet.Namespace) Then

Me.Namespace = table.Namespace

End If

Me.Prefix = table.Prefix

Me.MinimumCapacity = table.MinimumCapacity

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Sub New(ByVal info As Global.System.Runtime.Serialization.SerializationInfo, ByVal context As Global.System.Runtime.Serialization.StreamingContext)

MyBase.New(info, context)

Me.InitVars

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public ReadOnly Property IDColumn() As Global.System.Data.DataColumn

Get

Return Me.columnID

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public ReadOnly Property itemsColumn() As Global.System.Data.DataColumn

Get

Return Me.columnitems

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public ReadOnly Property quantityColumn() As Global.System.Data.DataColumn

Get

Return Me.columnquantity

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Browsable(false)> \_

Public ReadOnly Property Count() As Integer

Get

Return Me.Rows.Count

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Default ReadOnly Property Item(ByVal index As Integer) As stockRow

Get

Return CType(Me.Rows(index),stockRow)

End Get

End Property

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Event stockRowChanging As stockRowChangeEventHandler

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Event stockRowChanged As stockRowChangeEventHandler

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Event stockRowDeleting As stockRowChangeEventHandler

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Event stockRowDeleted As stockRowChangeEventHandler

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Overloads Sub AddstockRow(ByVal row As stockRow)

Me.Rows.Add(row)

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Overloads Function AddstockRow(ByVal items As String, ByVal quantity As Integer) As stockRow

Dim rowstockRow As stockRow = CType(Me.NewRow,stockRow)

Dim columnValuesArray() As Object = New Object() {Nothing, items, quantity}

rowstockRow.ItemArray = columnValuesArray

Me.Rows.Add(rowstockRow)

Return rowstockRow

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Function FindByID(ByVal ID As Integer) As stockRow

Return CType(Me.Rows.Find(New Object() {ID}),stockRow)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Overrides Function Clone() As Global.System.Data.DataTable

Dim cln As stockDataTable = CType(MyBase.Clone,stockDataTable)

cln.InitVars

Return cln

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Function CreateInstance() As Global.System.Data.DataTable

Return New stockDataTable()

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Sub InitVars()

Me.columnID = MyBase.Columns("ID")

Me.columnitems = MyBase.Columns("items")

Me.columnquantity = MyBase.Columns("quantity")

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Sub InitClass()

Me.columnID = New Global.System.Data.DataColumn("ID", GetType(Integer), Nothing, Global.System.Data.MappingType.Element)

MyBase.Columns.Add(Me.columnID)

Me.columnitems = New Global.System.Data.DataColumn("items", GetType(String), Nothing, Global.System.Data.MappingType.Element)

MyBase.Columns.Add(Me.columnitems)

Me.columnquantity = New Global.System.Data.DataColumn("quantity", GetType(Integer), Nothing, Global.System.Data.MappingType.Element)

MyBase.Columns.Add(Me.columnquantity)

Me.Constraints.Add(New Global.System.Data.UniqueConstraint("Constraint1", New Global.System.Data.DataColumn() {Me.columnID}, true))

Me.columnID.AutoIncrement = true

Me.columnID.AutoIncrementSeed = -1

Me.columnID.AutoIncrementStep = -1

Me.columnID.AllowDBNull = false

Me.columnID.Unique = true

Me.columnitems.MaxLength = 255

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Function NewstockRow() As stockRow

Return CType(Me.NewRow,stockRow)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Function NewRowFromBuilder(ByVal builder As Global.System.Data.DataRowBuilder) As Global.System.Data.DataRow

Return New stockRow(builder)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Function GetRowType() As Global.System.Type

Return GetType(stockRow)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Sub OnRowChanged(ByVal e As Global.System.Data.DataRowChangeEventArgs)

MyBase.OnRowChanged(e)

If (Not (Me.stockRowChangedEvent) Is Nothing) Then

RaiseEvent stockRowChanged(Me, New stockRowChangeEvent(CType(e.Row,stockRow), e.Action))

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Sub OnRowChanging(ByVal e As Global.System.Data.DataRowChangeEventArgs)

MyBase.OnRowChanging(e)

If (Not (Me.stockRowChangingEvent) Is Nothing) Then

RaiseEvent stockRowChanging(Me, New stockRowChangeEvent(CType(e.Row,stockRow), e.Action))

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Sub OnRowDeleted(ByVal e As Global.System.Data.DataRowChangeEventArgs)

MyBase.OnRowDeleted(e)

If (Not (Me.stockRowDeletedEvent) Is Nothing) Then

RaiseEvent stockRowDeleted(Me, New stockRowChangeEvent(CType(e.Row,stockRow), e.Action))

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overrides Sub OnRowDeleting(ByVal e As Global.System.Data.DataRowChangeEventArgs)

MyBase.OnRowDeleting(e)

If (Not (Me.stockRowDeletingEvent) Is Nothing) Then

RaiseEvent stockRowDeleting(Me, New stockRowChangeEvent(CType(e.Row,stockRow), e.Action))

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub RemovestockRow(ByVal row As stockRow)

Me.Rows.Remove(row)

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Shared Function GetTypedTableSchema(ByVal xs As Global.System.Xml.Schema.XmlSchemaSet) As Global.System.Xml.Schema.XmlSchemaComplexType

Dim type As Global.System.Xml.Schema.XmlSchemaComplexType = New Global.System.Xml.Schema.XmlSchemaComplexType()

Dim sequence As Global.System.Xml.Schema.XmlSchemaSequence = New Global.System.Xml.Schema.XmlSchemaSequence()

Dim ds As InvoiceDataSet = New InvoiceDataSet()

Dim any1 As Global.System.Xml.Schema.XmlSchemaAny = New Global.System.Xml.Schema.XmlSchemaAny()

any1.Namespace = "http://www.w3.org/2001/XMLSchema"

any1.MinOccurs = New Decimal(0)

any1.MaxOccurs = Decimal.MaxValue

any1.ProcessContents = Global.System.Xml.Schema.XmlSchemaContentProcessing.Lax

sequence.Items.Add(any1)

Dim any2 As Global.System.Xml.Schema.XmlSchemaAny = New Global.System.Xml.Schema.XmlSchemaAny()

any2.Namespace = "urn:schemas-microsoft-com:xml-diffgram-v1"

any2.MinOccurs = New Decimal(1)

any2.ProcessContents = Global.System.Xml.Schema.XmlSchemaContentProcessing.Lax

sequence.Items.Add(any2)

Dim attribute1 As Global.System.Xml.Schema.XmlSchemaAttribute = New Global.System.Xml.Schema.XmlSchemaAttribute()

attribute1.Name = "namespace"

attribute1.FixedValue = ds.Namespace

type.Attributes.Add(attribute1)

Dim attribute2 As Global.System.Xml.Schema.XmlSchemaAttribute = New Global.System.Xml.Schema.XmlSchemaAttribute()

attribute2.Name = "tableTypeName"

attribute2.FixedValue = "stockDataTable"

type.Attributes.Add(attribute2)

type.Particle = sequence

Dim dsSchema As Global.System.Xml.Schema.XmlSchema = ds.GetSchemaSerializable

If xs.Contains(dsSchema.TargetNamespace) Then

Dim s1 As Global.System.IO.MemoryStream = New Global.System.IO.MemoryStream()

Dim s2 As Global.System.IO.MemoryStream = New Global.System.IO.MemoryStream()

Try

Dim schema As Global.System.Xml.Schema.XmlSchema = Nothing

dsSchema.Write(s1)

Dim schemas As Global.System.Collections.IEnumerator = xs.Schemas(dsSchema.TargetNamespace).GetEnumerator

Do While schemas.MoveNext

schema = CType(schemas.Current,Global.System.Xml.Schema.XmlSchema)

s2.SetLength(0)

schema.Write(s2)

If (s1.Length = s2.Length) Then

s1.Position = 0

s2.Position = 0

Do While ((s1.Position <> s1.Length) \_

AndAlso (s1.ReadByte = s2.ReadByte))

Loop

If (s1.Position = s1.Length) Then

Return type

End If

End If

Loop

Finally

If (Not (s1) Is Nothing) Then

s1.Close

End If

If (Not (s2) Is Nothing) Then

s2.Close

End If

End Try

End If

xs.Add(dsSchema)

Return type

End Function

End Class

'''<summary>

'''Represents strongly named DataRow class.

'''</summary>

Partial Public Class adminRow

Inherits Global.System.Data.DataRow

Private tableadmin As adminDataTable

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Sub New(ByVal rb As Global.System.Data.DataRowBuilder)

MyBase.New(rb)

Me.tableadmin = CType(Me.Table,adminDataTable)

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Property ID() As Integer

Get

Return CType(Me(Me.tableadmin.IDColumn),Integer)

End Get

Set

Me(Me.tableadmin.IDColumn) = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Property username() As String

Get

Try

Return CType(Me(Me.tableadmin.usernameColumn),String)

Catch e As Global.System.InvalidCastException

Throw New Global.System.Data.StrongTypingException("The value for column 'username' in table 'admin' is DBNull.", e)

End Try

End Get

Set

Me(Me.tableadmin.usernameColumn) = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Property password() As String

Get

Try

Return CType(Me(Me.tableadmin.passwordColumn),String)

Catch e As Global.System.InvalidCastException

Throw New Global.System.Data.StrongTypingException("The value for column 'password' in table 'admin' is DBNull.", e)

End Try

End Get

Set

Me(Me.tableadmin.passwordColumn) = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Function IsusernameNull() As Boolean

Return Me.IsNull(Me.tableadmin.usernameColumn)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub SetusernameNull()

Me(Me.tableadmin.usernameColumn) = Global.System.Convert.DBNull

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Function IspasswordNull() As Boolean

Return Me.IsNull(Me.tableadmin.passwordColumn)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub SetpasswordNull()

Me(Me.tableadmin.passwordColumn) = Global.System.Convert.DBNull

End Sub

End Class

'''<summary>

'''Represents strongly named DataRow class.

'''</summary>

Partial Public Class stockRow

Inherits Global.System.Data.DataRow

Private tablestock As stockDataTable

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Sub New(ByVal rb As Global.System.Data.DataRowBuilder)

MyBase.New(rb)

Me.tablestock = CType(Me.Table,stockDataTable)

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Property ID() As Integer

Get

Return CType(Me(Me.tablestock.IDColumn),Integer)

End Get

Set

Me(Me.tablestock.IDColumn) = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Property items() As String

Get

Try

Return CType(Me(Me.tablestock.itemsColumn),String)

Catch e As Global.System.InvalidCastException

Throw New Global.System.Data.StrongTypingException("The value for column 'items' in table 'stock' is DBNull.", e)

End Try

End Get

Set

Me(Me.tablestock.itemsColumn) = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Property quantity() As Integer

Get

Try

Return CType(Me(Me.tablestock.quantityColumn),Integer)

Catch e As Global.System.InvalidCastException

Throw New Global.System.Data.StrongTypingException("The value for column 'quantity' in table 'stock' is DBNull.", e)

End Try

End Get

Set

Me(Me.tablestock.quantityColumn) = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Function IsitemsNull() As Boolean

Return Me.IsNull(Me.tablestock.itemsColumn)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub SetitemsNull()

Me(Me.tablestock.itemsColumn) = Global.System.Convert.DBNull

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Function IsquantityNull() As Boolean

Return Me.IsNull(Me.tablestock.quantityColumn)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub SetquantityNull()

Me(Me.tablestock.quantityColumn) = Global.System.Convert.DBNull

End Sub

End Class

'''<summary>

'''Row event argument class

'''</summary>

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Class adminRowChangeEvent

Inherits Global.System.EventArgs

Private eventRow As adminRow

Private eventAction As Global.System.Data.DataRowAction

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub New(ByVal row As adminRow, ByVal action As Global.System.Data.DataRowAction)

MyBase.New

Me.eventRow = row

Me.eventAction = action

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public ReadOnly Property Row() As adminRow

Get

Return Me.eventRow

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public ReadOnly Property Action() As Global.System.Data.DataRowAction

Get

Return Me.eventAction

End Get

End Property

End Class

'''<summary>

'''Row event argument class

'''</summary>

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Class stockRowChangeEvent

Inherits Global.System.EventArgs

Private eventRow As stockRow

Private eventAction As Global.System.Data.DataRowAction

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub New(ByVal row As stockRow, ByVal action As Global.System.Data.DataRowAction)

MyBase.New

Me.eventRow = row

Me.eventAction = action

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public ReadOnly Property Row() As stockRow

Get

Return Me.eventRow

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public ReadOnly Property Action() As Global.System.Data.DataRowAction

Get

Return Me.eventAction

End Get

End Property

End Class

End Class

Namespace InvoiceDataSetTableAdapters

'''<summary>

'''Represents the connection and commands used to retrieve and save data.

'''</summary>

<Global.System.ComponentModel.DesignerCategoryAttribute("code"), \_

Global.System.ComponentModel.ToolboxItem(true), \_

Global.System.ComponentModel.DataObjectAttribute(true), \_

Global.System.ComponentModel.DesignerAttribute("Microsoft.VSDesigner.DataSource.Design.TableAdapterDesigner, Microsoft.VSDesigner"& \_

", Version=10.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")> \_

Partial Public Class adminTableAdapter

Inherits Global.System.ComponentModel.Component

Private WithEvents \_adapter As Global.System.Data.OleDb.OleDbDataAdapter

Private \_connection As Global.System.Data.OleDb.OleDbConnection

Private \_transaction As Global.System.Data.OleDb.OleDbTransaction

Private \_commandCollection() As Global.System.Data.OleDb.OleDbCommand

Private \_clearBeforeFill As Boolean

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub New()

MyBase.New

Me.ClearBeforeFill = true

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Friend ReadOnly Property Adapter() As Global.System.Data.OleDb.OleDbDataAdapter

Get

If (Me.\_adapter Is Nothing) Then

Me.InitAdapter

End If

Return Me.\_adapter

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Property Connection() As Global.System.Data.OleDb.OleDbConnection

Get

If (Me.\_connection Is Nothing) Then

Me.InitConnection

End If

Return Me.\_connection

End Get

Set

Me.\_connection = value

If (Not (Me.Adapter.InsertCommand) Is Nothing) Then

Me.Adapter.InsertCommand.Connection = value

End If

If (Not (Me.Adapter.DeleteCommand) Is Nothing) Then

Me.Adapter.DeleteCommand.Connection = value

End If

If (Not (Me.Adapter.UpdateCommand) Is Nothing) Then

Me.Adapter.UpdateCommand.Connection = value

End If

Dim i As Integer = 0

Do While (i < Me.CommandCollection.Length)

If (Not (Me.CommandCollection(i)) Is Nothing) Then

CType(Me.CommandCollection(i),Global.System.Data.OleDb.OleDbCommand).Connection = value

End If

i = (i + 1)

Loop

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Property Transaction() As Global.System.Data.OleDb.OleDbTransaction

Get

Return Me.\_transaction

End Get

Set

Me.\_transaction = value

Dim i As Integer = 0

Do While (i < Me.CommandCollection.Length)

Me.CommandCollection(i).Transaction = Me.\_transaction

i = (i + 1)

Loop

If ((Not (Me.Adapter) Is Nothing) \_

AndAlso (Not (Me.Adapter.DeleteCommand) Is Nothing)) Then

Me.Adapter.DeleteCommand.Transaction = Me.\_transaction

End If

If ((Not (Me.Adapter) Is Nothing) \_

AndAlso (Not (Me.Adapter.InsertCommand) Is Nothing)) Then

Me.Adapter.InsertCommand.Transaction = Me.\_transaction

End If

If ((Not (Me.Adapter) Is Nothing) \_

AndAlso (Not (Me.Adapter.UpdateCommand) Is Nothing)) Then

Me.Adapter.UpdateCommand.Transaction = Me.\_transaction

End If

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected ReadOnly Property CommandCollection() As Global.System.Data.OleDb.OleDbCommand()

Get

If (Me.\_commandCollection Is Nothing) Then

Me.InitCommandCollection

End If

Return Me.\_commandCollection

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Property ClearBeforeFill() As Boolean

Get

Return Me.\_clearBeforeFill

End Get

Set

Me.\_clearBeforeFill = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Sub InitAdapter()

Me.\_adapter = New Global.System.Data.OleDb.OleDbDataAdapter()

Dim tableMapping As Global.System.Data.Common.DataTableMapping = New Global.System.Data.Common.DataTableMapping()

tableMapping.SourceTable = "Table"

tableMapping.DataSetTable = "admin"

tableMapping.ColumnMappings.Add("ID", "ID")

tableMapping.ColumnMappings.Add("username", "username")

tableMapping.ColumnMappings.Add("password", "password")

Me.\_adapter.TableMappings.Add(tableMapping)

Me.\_adapter.DeleteCommand = New Global.System.Data.OleDb.OleDbCommand()

Me.\_adapter.DeleteCommand.Connection = Me.Connection

Me.\_adapter.DeleteCommand.CommandText = "DELETE FROM `admin` WHERE ((`ID` = ?) AND ((? = 1 AND `username` IS NULL) OR (`us"& \_

"ername` = ?)) AND ((? = 1 AND `password` IS NULL) OR (`password` = ?)))"

Me.\_adapter.DeleteCommand.CommandType = Global.System.Data.CommandType.Text

Me.\_adapter.DeleteCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_ID", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "ID", Global.System.Data.DataRowVersion.Original, false, Nothing))

Me.\_adapter.DeleteCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("IsNull\_username", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "username", Global.System.Data.DataRowVersion.Original, true, Nothing))

Me.\_adapter.DeleteCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_username", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "username", Global.System.Data.DataRowVersion.Original, false, Nothing))

Me.\_adapter.DeleteCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("IsNull\_password", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "password", Global.System.Data.DataRowVersion.Original, true, Nothing))

Me.\_adapter.DeleteCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_password", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "password", Global.System.Data.DataRowVersion.Original, false, Nothing))

Me.\_adapter.InsertCommand = New Global.System.Data.OleDb.OleDbCommand()

Me.\_adapter.InsertCommand.Connection = Me.Connection

Me.\_adapter.InsertCommand.CommandText = "INSERT INTO `admin` (`username`, `password`) VALUES (?, ?)"

Me.\_adapter.InsertCommand.CommandType = Global.System.Data.CommandType.Text

Me.\_adapter.InsertCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("username", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "username", Global.System.Data.DataRowVersion.Current, false, Nothing))

Me.\_adapter.InsertCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("password", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "password", Global.System.Data.DataRowVersion.Current, false, Nothing))

Me.\_adapter.UpdateCommand = New Global.System.Data.OleDb.OleDbCommand()

Me.\_adapter.UpdateCommand.Connection = Me.Connection

Me.\_adapter.UpdateCommand.CommandText = "UPDATE `admin` SET `username` = ?, `password` = ? WHERE ((`ID` = ?) AND ((? = 1 A"& \_

"ND `username` IS NULL) OR (`username` = ?)) AND ((? = 1 AND `password` IS NULL) "& \_

"OR (`password` = ?)))"

Me.\_adapter.UpdateCommand.CommandType = Global.System.Data.CommandType.Text

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("username", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "username", Global.System.Data.DataRowVersion.Current, false, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("password", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "password", Global.System.Data.DataRowVersion.Current, false, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_ID", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "ID", Global.System.Data.DataRowVersion.Original, false, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("IsNull\_username", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "username", Global.System.Data.DataRowVersion.Original, true, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_username", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "username", Global.System.Data.DataRowVersion.Original, false, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("IsNull\_password", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "password", Global.System.Data.DataRowVersion.Original, true, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_password", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "password", Global.System.Data.DataRowVersion.Original, false, Nothing))

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Sub InitConnection()

Me.\_connection = New Global.System.Data.OleDb.OleDbConnection()

Me.\_connection.ConnectionString = Global.Invoice.My.MySettings.Default.InvoiceConnectionString

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Sub InitCommandCollection()

Me.\_commandCollection = New Global.System.Data.OleDb.OleDbCommand(0) {}

Me.\_commandCollection(0) = New Global.System.Data.OleDb.OleDbCommand()

Me.\_commandCollection(0).Connection = Me.Connection

Me.\_commandCollection(0).CommandText = "SELECT ID, username, [password] FROM admin"

Me.\_commandCollection(0).CommandType = Global.System.Data.CommandType.Text

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter"), \_

Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.ComponentModel.DataObjectMethodType.Fill, true)> \_

Public Overloads Overridable Function Fill(ByVal dataTable As InvoiceDataSet.adminDataTable) As Integer

Me.Adapter.SelectCommand = Me.CommandCollection(0)

If (Me.ClearBeforeFill = true) Then

dataTable.Clear

End If

Dim returnValue As Integer = Me.Adapter.Fill(dataTable)

Return returnValue

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter"), \_

Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.ComponentModel.DataObjectMethodType.[Select], true)> \_

Public Overloads Overridable Function GetData() As InvoiceDataSet.adminDataTable

Me.Adapter.SelectCommand = Me.CommandCollection(0)

Dim dataTable As InvoiceDataSet.adminDataTable = New InvoiceDataSet.adminDataTable()

Me.Adapter.Fill(dataTable)

Return dataTable

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")> \_

Public Overloads Overridable Function Update(ByVal dataTable As InvoiceDataSet.adminDataTable) As Integer

Return Me.Adapter.Update(dataTable)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")> \_

Public Overloads Overridable Function Update(ByVal dataSet As InvoiceDataSet) As Integer

Return Me.Adapter.Update(dataSet, "admin")

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")> \_

Public Overloads Overridable Function Update(ByVal dataRow As Global.System.Data.DataRow) As Integer

Return Me.Adapter.Update(New Global.System.Data.DataRow() {dataRow})

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")> \_

Public Overloads Overridable Function Update(ByVal dataRows() As Global.System.Data.DataRow) As Integer

Return Me.Adapter.Update(dataRows)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter"), \_

Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.ComponentModel.DataObjectMethodType.Delete, true)> \_

Public Overloads Overridable Function Delete(ByVal Original\_ID As Integer, ByVal Original\_username As String, ByVal Original\_password As String) As Integer

Me.Adapter.DeleteCommand.Parameters(0).Value = CType(Original\_ID,Integer)

If (Original\_username Is Nothing) Then

Me.Adapter.DeleteCommand.Parameters(1).Value = CType(1,Object)

Me.Adapter.DeleteCommand.Parameters(2).Value = Global.System.DBNull.Value

Else

Me.Adapter.DeleteCommand.Parameters(1).Value = CType(0,Object)

Me.Adapter.DeleteCommand.Parameters(2).Value = CType(Original\_username,String)

End If

If (Original\_password Is Nothing) Then

Me.Adapter.DeleteCommand.Parameters(3).Value = CType(1,Object)

Me.Adapter.DeleteCommand.Parameters(4).Value = Global.System.DBNull.Value

Else

Me.Adapter.DeleteCommand.Parameters(3).Value = CType(0,Object)

Me.Adapter.DeleteCommand.Parameters(4).Value = CType(Original\_password,String)

End If

Dim previousConnectionState As Global.System.Data.ConnectionState = Me.Adapter.DeleteCommand.Connection.State

If ((Me.Adapter.DeleteCommand.Connection.State And Global.System.Data.ConnectionState.Open) \_

<> Global.System.Data.ConnectionState.Open) Then

Me.Adapter.DeleteCommand.Connection.Open

End If

Try

Dim returnValue As Integer = Me.Adapter.DeleteCommand.ExecuteNonQuery

Return returnValue

Finally

If (previousConnectionState = Global.System.Data.ConnectionState.Closed) Then

Me.Adapter.DeleteCommand.Connection.Close

End If

End Try

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter"), \_

Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.ComponentModel.DataObjectMethodType.Insert, true)> \_

Public Overloads Overridable Function Insert(ByVal username As String, ByVal password As String) As Integer

If (username Is Nothing) Then

Me.Adapter.InsertCommand.Parameters(0).Value = Global.System.DBNull.Value

Else

Me.Adapter.InsertCommand.Parameters(0).Value = CType(username,String)

End If

If (password Is Nothing) Then

Me.Adapter.InsertCommand.Parameters(1).Value = Global.System.DBNull.Value

Else

Me.Adapter.InsertCommand.Parameters(1).Value = CType(password,String)

End If

Dim previousConnectionState As Global.System.Data.ConnectionState = Me.Adapter.InsertCommand.Connection.State

If ((Me.Adapter.InsertCommand.Connection.State And Global.System.Data.ConnectionState.Open) \_

<> Global.System.Data.ConnectionState.Open) Then

Me.Adapter.InsertCommand.Connection.Open

End If

Try

Dim returnValue As Integer = Me.Adapter.InsertCommand.ExecuteNonQuery

Return returnValue

Finally

If (previousConnectionState = Global.System.Data.ConnectionState.Closed) Then

Me.Adapter.InsertCommand.Connection.Close

End If

End Try

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter"), \_

Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.ComponentModel.DataObjectMethodType.Update, true)> \_

Public Overloads Overridable Function Update(ByVal username As String, ByVal password As String, ByVal Original\_ID As Integer, ByVal Original\_username As String, ByVal Original\_password As String) As Integer

If (username Is Nothing) Then

Me.Adapter.UpdateCommand.Parameters(0).Value = Global.System.DBNull.Value

Else

Me.Adapter.UpdateCommand.Parameters(0).Value = CType(username,String)

End If

If (password Is Nothing) Then

Me.Adapter.UpdateCommand.Parameters(1).Value = Global.System.DBNull.Value

Else

Me.Adapter.UpdateCommand.Parameters(1).Value = CType(password,String)

End If

Me.Adapter.UpdateCommand.Parameters(2).Value = CType(Original\_ID,Integer)

If (Original\_username Is Nothing) Then

Me.Adapter.UpdateCommand.Parameters(3).Value = CType(1,Object)

Me.Adapter.UpdateCommand.Parameters(4).Value = Global.System.DBNull.Value

Else

Me.Adapter.UpdateCommand.Parameters(3).Value = CType(0,Object)

Me.Adapter.UpdateCommand.Parameters(4).Value = CType(Original\_username,String)

End If

If (Original\_password Is Nothing) Then

Me.Adapter.UpdateCommand.Parameters(5).Value = CType(1,Object)

Me.Adapter.UpdateCommand.Parameters(6).Value = Global.System.DBNull.Value

Else

Me.Adapter.UpdateCommand.Parameters(5).Value = CType(0,Object)

Me.Adapter.UpdateCommand.Parameters(6).Value = CType(Original\_password,String)

End If

Dim previousConnectionState As Global.System.Data.ConnectionState = Me.Adapter.UpdateCommand.Connection.State

If ((Me.Adapter.UpdateCommand.Connection.State And Global.System.Data.ConnectionState.Open) \_

<> Global.System.Data.ConnectionState.Open) Then

Me.Adapter.UpdateCommand.Connection.Open

End If

Try

Dim returnValue As Integer = Me.Adapter.UpdateCommand.ExecuteNonQuery

Return returnValue

Finally

If (previousConnectionState = Global.System.Data.ConnectionState.Closed) Then

Me.Adapter.UpdateCommand.Connection.Close

End If

End Try

End Function

End Class

'''<summary>

'''Represents the connection and commands used to retrieve and save data.

'''</summary>

<Global.System.ComponentModel.DesignerCategoryAttribute("code"), \_

Global.System.ComponentModel.ToolboxItem(true), \_

Global.System.ComponentModel.DataObjectAttribute(true), \_

Global.System.ComponentModel.DesignerAttribute("Microsoft.VSDesigner.DataSource.Design.TableAdapterDesigner, Microsoft.VSDesigner"& \_

", Version=10.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")> \_

Partial Public Class stockTableAdapter

Inherits Global.System.ComponentModel.Component

Private WithEvents \_adapter As Global.System.Data.OleDb.OleDbDataAdapter

Private \_connection As Global.System.Data.OleDb.OleDbConnection

Private \_transaction As Global.System.Data.OleDb.OleDbTransaction

Private \_commandCollection() As Global.System.Data.OleDb.OleDbCommand

Private \_clearBeforeFill As Boolean

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Sub New()

MyBase.New

Me.ClearBeforeFill = true

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Friend ReadOnly Property Adapter() As Global.System.Data.OleDb.OleDbDataAdapter

Get

If (Me.\_adapter Is Nothing) Then

Me.InitAdapter

End If

Return Me.\_adapter

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Property Connection() As Global.System.Data.OleDb.OleDbConnection

Get

If (Me.\_connection Is Nothing) Then

Me.InitConnection

End If

Return Me.\_connection

End Get

Set

Me.\_connection = value

If (Not (Me.Adapter.InsertCommand) Is Nothing) Then

Me.Adapter.InsertCommand.Connection = value

End If

If (Not (Me.Adapter.DeleteCommand) Is Nothing) Then

Me.Adapter.DeleteCommand.Connection = value

End If

If (Not (Me.Adapter.UpdateCommand) Is Nothing) Then

Me.Adapter.UpdateCommand.Connection = value

End If

Dim i As Integer = 0

Do While (i < Me.CommandCollection.Length)

If (Not (Me.CommandCollection(i)) Is Nothing) Then

CType(Me.CommandCollection(i),Global.System.Data.OleDb.OleDbCommand).Connection = value

End If

i = (i + 1)

Loop

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Property Transaction() As Global.System.Data.OleDb.OleDbTransaction

Get

Return Me.\_transaction

End Get

Set

Me.\_transaction = value

Dim i As Integer = 0

Do While (i < Me.CommandCollection.Length)

Me.CommandCollection(i).Transaction = Me.\_transaction

i = (i + 1)

Loop

If ((Not (Me.Adapter) Is Nothing) \_

AndAlso (Not (Me.Adapter.DeleteCommand) Is Nothing)) Then

Me.Adapter.DeleteCommand.Transaction = Me.\_transaction

End If

If ((Not (Me.Adapter) Is Nothing) \_

AndAlso (Not (Me.Adapter.InsertCommand) Is Nothing)) Then

Me.Adapter.InsertCommand.Transaction = Me.\_transaction

End If

If ((Not (Me.Adapter) Is Nothing) \_

AndAlso (Not (Me.Adapter.UpdateCommand) Is Nothing)) Then

Me.Adapter.UpdateCommand.Transaction = Me.\_transaction

End If

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected ReadOnly Property CommandCollection() As Global.System.Data.OleDb.OleDbCommand()

Get

If (Me.\_commandCollection Is Nothing) Then

Me.InitCommandCollection

End If

Return Me.\_commandCollection

End Get

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Property ClearBeforeFill() As Boolean

Get

Return Me.\_clearBeforeFill

End Get

Set

Me.\_clearBeforeFill = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Sub InitAdapter()

Me.\_adapter = New Global.System.Data.OleDb.OleDbDataAdapter()

Dim tableMapping As Global.System.Data.Common.DataTableMapping = New Global.System.Data.Common.DataTableMapping()

tableMapping.SourceTable = "Table"

tableMapping.DataSetTable = "stock"

tableMapping.ColumnMappings.Add("ID", "ID")

tableMapping.ColumnMappings.Add("items", "items")

tableMapping.ColumnMappings.Add("quantity", "quantity")

Me.\_adapter.TableMappings.Add(tableMapping)

Me.\_adapter.DeleteCommand = New Global.System.Data.OleDb.OleDbCommand()

Me.\_adapter.DeleteCommand.Connection = Me.Connection

Me.\_adapter.DeleteCommand.CommandText = "DELETE FROM `stock` WHERE ((`ID` = ?) AND ((? = 1 AND `items` IS NULL) OR (`items"& \_

"` = ?)) AND ((? = 1 AND `quantity` IS NULL) OR (`quantity` = ?)))"

Me.\_adapter.DeleteCommand.CommandType = Global.System.Data.CommandType.Text

Me.\_adapter.DeleteCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_ID", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "ID", Global.System.Data.DataRowVersion.Original, false, Nothing))

Me.\_adapter.DeleteCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("IsNull\_items", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "items", Global.System.Data.DataRowVersion.Original, true, Nothing))

Me.\_adapter.DeleteCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_items", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "items", Global.System.Data.DataRowVersion.Original, false, Nothing))

Me.\_adapter.DeleteCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("IsNull\_quantity", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "quantity", Global.System.Data.DataRowVersion.Original, true, Nothing))

Me.\_adapter.DeleteCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_quantity", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "quantity", Global.System.Data.DataRowVersion.Original, false, Nothing))

Me.\_adapter.InsertCommand = New Global.System.Data.OleDb.OleDbCommand()

Me.\_adapter.InsertCommand.Connection = Me.Connection

Me.\_adapter.InsertCommand.CommandText = "INSERT INTO `stock` (`items`, `quantity`) VALUES (?, ?)"

Me.\_adapter.InsertCommand.CommandType = Global.System.Data.CommandType.Text

Me.\_adapter.InsertCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("items", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "items", Global.System.Data.DataRowVersion.Current, false, Nothing))

Me.\_adapter.InsertCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("quantity", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "quantity", Global.System.Data.DataRowVersion.Current, false, Nothing))

Me.\_adapter.UpdateCommand = New Global.System.Data.OleDb.OleDbCommand()

Me.\_adapter.UpdateCommand.Connection = Me.Connection

Me.\_adapter.UpdateCommand.CommandText = "UPDATE `stock` SET `items` = ?, `quantity` = ? WHERE ((`ID` = ?) AND ((? = 1 AND "& \_

"`items` IS NULL) OR (`items` = ?)) AND ((? = 1 AND `quantity` IS NULL) OR (`quan"& \_

"tity` = ?)))"

Me.\_adapter.UpdateCommand.CommandType = Global.System.Data.CommandType.Text

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("items", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "items", Global.System.Data.DataRowVersion.Current, false, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("quantity", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "quantity", Global.System.Data.DataRowVersion.Current, false, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_ID", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "ID", Global.System.Data.DataRowVersion.Original, false, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("IsNull\_items", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "items", Global.System.Data.DataRowVersion.Original, true, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_items", Global.System.Data.OleDb.OleDbType.VarWChar, 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "items", Global.System.Data.DataRowVersion.Original, false, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("IsNull\_quantity", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "quantity", Global.System.Data.DataRowVersion.Original, true, Nothing))

Me.\_adapter.UpdateCommand.Parameters.Add(New Global.System.Data.OleDb.OleDbParameter("Original\_quantity", Global.System.Data.OleDb.OleDbType.[Integer], 0, Global.System.Data.ParameterDirection.Input, CType(0,Byte), CType(0,Byte), "quantity", Global.System.Data.DataRowVersion.Original, false, Nothing))

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Sub InitConnection()

Me.\_connection = New Global.System.Data.OleDb.OleDbConnection()

Me.\_connection.ConnectionString = Global.Invoice.My.MySettings.Default.InvoiceConnectionString

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Sub InitCommandCollection()

Me.\_commandCollection = New Global.System.Data.OleDb.OleDbCommand(0) {}

Me.\_commandCollection(0) = New Global.System.Data.OleDb.OleDbCommand()

Me.\_commandCollection(0).Connection = Me.Connection

Me.\_commandCollection(0).CommandText = "SELECT ID, items, quantity FROM stock"

Me.\_commandCollection(0).CommandType = Global.System.Data.CommandType.Text

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter"), \_

Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.ComponentModel.DataObjectMethodType.Fill, true)> \_

Public Overloads Overridable Function Fill(ByVal dataTable As InvoiceDataSet.stockDataTable) As Integer

Me.Adapter.SelectCommand = Me.CommandCollection(0)

If (Me.ClearBeforeFill = true) Then

dataTable.Clear

End If

Dim returnValue As Integer = Me.Adapter.Fill(dataTable)

Return returnValue

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter"), \_

Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.ComponentModel.DataObjectMethodType.[Select], true)> \_

Public Overloads Overridable Function GetData() As InvoiceDataSet.stockDataTable

Me.Adapter.SelectCommand = Me.CommandCollection(0)

Dim dataTable As InvoiceDataSet.stockDataTable = New InvoiceDataSet.stockDataTable()

Me.Adapter.Fill(dataTable)

Return dataTable

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")> \_

Public Overloads Overridable Function Update(ByVal dataTable As InvoiceDataSet.stockDataTable) As Integer

Return Me.Adapter.Update(dataTable)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")> \_

Public Overloads Overridable Function Update(ByVal dataSet As InvoiceDataSet) As Integer

Return Me.Adapter.Update(dataSet, "stock")

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")> \_

Public Overloads Overridable Function Update(ByVal dataRow As Global.System.Data.DataRow) As Integer

Return Me.Adapter.Update(New Global.System.Data.DataRow() {dataRow})

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")> \_

Public Overloads Overridable Function Update(ByVal dataRows() As Global.System.Data.DataRow) As Integer

Return Me.Adapter.Update(dataRows)

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter"), \_

Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.ComponentModel.DataObjectMethodType.Delete, true)> \_

Public Overloads Overridable Function Delete(ByVal Original\_ID As Integer, ByVal Original\_items As String, ByVal Original\_quantity As Global.System.Nullable(Of Integer)) As Integer

Me.Adapter.DeleteCommand.Parameters(0).Value = CType(Original\_ID,Integer)

If (Original\_items Is Nothing) Then

Me.Adapter.DeleteCommand.Parameters(1).Value = CType(1,Object)

Me.Adapter.DeleteCommand.Parameters(2).Value = Global.System.DBNull.Value

Else

Me.Adapter.DeleteCommand.Parameters(1).Value = CType(0,Object)

Me.Adapter.DeleteCommand.Parameters(2).Value = CType(Original\_items,String)

End If

If (Original\_quantity.HasValue = true) Then

Me.Adapter.DeleteCommand.Parameters(3).Value = CType(0,Object)

Me.Adapter.DeleteCommand.Parameters(4).Value = CType(Original\_quantity.Value,Integer)

Else

Me.Adapter.DeleteCommand.Parameters(3).Value = CType(1,Object)

Me.Adapter.DeleteCommand.Parameters(4).Value = Global.System.DBNull.Value

End If

Dim previousConnectionState As Global.System.Data.ConnectionState = Me.Adapter.DeleteCommand.Connection.State

If ((Me.Adapter.DeleteCommand.Connection.State And Global.System.Data.ConnectionState.Open) \_

<> Global.System.Data.ConnectionState.Open) Then

Me.Adapter.DeleteCommand.Connection.Open

End If

Try

Dim returnValue As Integer = Me.Adapter.DeleteCommand.ExecuteNonQuery

Return returnValue

Finally

If (previousConnectionState = Global.System.Data.ConnectionState.Closed) Then

Me.Adapter.DeleteCommand.Connection.Close

End If

End Try

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter"), \_

Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.ComponentModel.DataObjectMethodType.Insert, true)> \_

Public Overloads Overridable Function Insert(ByVal items As String, ByVal quantity As Global.System.Nullable(Of Integer)) As Integer

If (items Is Nothing) Then

Me.Adapter.InsertCommand.Parameters(0).Value = Global.System.DBNull.Value

Else

Me.Adapter.InsertCommand.Parameters(0).Value = CType(items,String)

End If

If (quantity.HasValue = true) Then

Me.Adapter.InsertCommand.Parameters(1).Value = CType(quantity.Value,Integer)

Else

Me.Adapter.InsertCommand.Parameters(1).Value = Global.System.DBNull.Value

End If

Dim previousConnectionState As Global.System.Data.ConnectionState = Me.Adapter.InsertCommand.Connection.State

If ((Me.Adapter.InsertCommand.Connection.State And Global.System.Data.ConnectionState.Open) \_

<> Global.System.Data.ConnectionState.Open) Then

Me.Adapter.InsertCommand.Connection.Open

End If

Try

Dim returnValue As Integer = Me.Adapter.InsertCommand.ExecuteNonQuery

Return returnValue

Finally

If (previousConnectionState = Global.System.Data.ConnectionState.Closed) Then

Me.Adapter.InsertCommand.Connection.Close

End If

End Try

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter"), \_

Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.ComponentModel.DataObjectMethodType.Update, true)> \_

Public Overloads Overridable Function Update(ByVal items As String, ByVal quantity As Global.System.Nullable(Of Integer), ByVal Original\_ID As Integer, ByVal Original\_items As String, ByVal Original\_quantity As Global.System.Nullable(Of Integer)) As Integer

If (items Is Nothing) Then

Me.Adapter.UpdateCommand.Parameters(0).Value = Global.System.DBNull.Value

Else

Me.Adapter.UpdateCommand.Parameters(0).Value = CType(items,String)

End If

If (quantity.HasValue = true) Then

Me.Adapter.UpdateCommand.Parameters(1).Value = CType(quantity.Value,Integer)

Else

Me.Adapter.UpdateCommand.Parameters(1).Value = Global.System.DBNull.Value

End If

Me.Adapter.UpdateCommand.Parameters(2).Value = CType(Original\_ID,Integer)

If (Original\_items Is Nothing) Then

Me.Adapter.UpdateCommand.Parameters(3).Value = CType(1,Object)

Me.Adapter.UpdateCommand.Parameters(4).Value = Global.System.DBNull.Value

Else

Me.Adapter.UpdateCommand.Parameters(3).Value = CType(0,Object)

Me.Adapter.UpdateCommand.Parameters(4).Value = CType(Original\_items,String)

End If

If (Original\_quantity.HasValue = true) Then

Me.Adapter.UpdateCommand.Parameters(5).Value = CType(0,Object)

Me.Adapter.UpdateCommand.Parameters(6).Value = CType(Original\_quantity.Value,Integer)

Else

Me.Adapter.UpdateCommand.Parameters(5).Value = CType(1,Object)

Me.Adapter.UpdateCommand.Parameters(6).Value = Global.System.DBNull.Value

End If

Dim previousConnectionState As Global.System.Data.ConnectionState = Me.Adapter.UpdateCommand.Connection.State

If ((Me.Adapter.UpdateCommand.Connection.State And Global.System.Data.ConnectionState.Open) \_

<> Global.System.Data.ConnectionState.Open) Then

Me.Adapter.UpdateCommand.Connection.Open

End If

Try

Dim returnValue As Integer = Me.Adapter.UpdateCommand.ExecuteNonQuery

Return returnValue

Finally

If (previousConnectionState = Global.System.Data.ConnectionState.Closed) Then

Me.Adapter.UpdateCommand.Connection.Close

End If

End Try

End Function

End Class

'''<summary>

'''TableAdapterManager is used to coordinate TableAdapters in the dataset to enable Hierarchical Update scenarios

'''</summary>

<Global.System.ComponentModel.DesignerCategoryAttribute("code"), \_

Global.System.ComponentModel.ToolboxItem(true), \_

Global.System.ComponentModel.DesignerAttribute("Microsoft.VSDesigner.DataSource.Design.TableAdapterManagerDesigner, Microsoft.VSD"& \_

"esigner, Version=10.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"), \_

Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapterManager")> \_

Partial Public Class TableAdapterManager

Inherits Global.System.ComponentModel.Component

Private \_updateOrder As UpdateOrderOption

Private \_adminTableAdapter As adminTableAdapter

Private \_stockTableAdapter As stockTableAdapter

Private \_backupDataSetBeforeUpdate As Boolean

Private \_connection As Global.System.Data.IDbConnection

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Property UpdateOrder() As UpdateOrderOption

Get

Return Me.\_updateOrder

End Get

Set

Me.\_updateOrder = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.EditorAttribute("Microsoft.VSDesigner.DataSource.Design.TableAdapterManagerPropertyEditor, Microso"& \_

"ft.VSDesigner, Version=10.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3"& \_

"a", "System.Drawing.Design.UITypeEditor")> \_

Public Property adminTableAdapter() As adminTableAdapter

Get

Return Me.\_adminTableAdapter

End Get

Set

Me.\_adminTableAdapter = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.EditorAttribute("Microsoft.VSDesigner.DataSource.Design.TableAdapterManagerPropertyEditor, Microso"& \_

"ft.VSDesigner, Version=10.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3"& \_

"a", "System.Drawing.Design.UITypeEditor")> \_

Public Property stockTableAdapter() As stockTableAdapter

Get

Return Me.\_stockTableAdapter

End Get

Set

Me.\_stockTableAdapter = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Property BackupDataSetBeforeUpdate() As Boolean

Get

Return Me.\_backupDataSetBeforeUpdate

End Get

Set

Me.\_backupDataSetBeforeUpdate = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Browsable(false)> \_

Public Property Connection() As Global.System.Data.IDbConnection

Get

If (Not (Me.\_connection) Is Nothing) Then

Return Me.\_connection

End If

If ((Not (Me.\_adminTableAdapter) Is Nothing) \_

AndAlso (Not (Me.\_adminTableAdapter.Connection) Is Nothing)) Then

Return Me.\_adminTableAdapter.Connection

End If

If ((Not (Me.\_stockTableAdapter) Is Nothing) \_

AndAlso (Not (Me.\_stockTableAdapter.Connection) Is Nothing)) Then

Return Me.\_stockTableAdapter.Connection

End If

Return Nothing

End Get

Set

Me.\_connection = value

End Set

End Property

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0"), \_

Global.System.ComponentModel.Browsable(false)> \_

Public ReadOnly Property TableAdapterInstanceCount() As Integer

Get

Dim count As Integer = 0

If (Not (Me.\_adminTableAdapter) Is Nothing) Then

count = (count + 1)

End If

If (Not (Me.\_stockTableAdapter) Is Nothing) Then

count = (count + 1)

End If

Return count

End Get

End Property

'''<summary>

'''Update rows in top-down order.

'''</summary>

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Function UpdateUpdatedRows(ByVal dataSet As InvoiceDataSet, ByVal allChangedRows As Global.System.Collections.Generic.List(Of Global.System.Data.DataRow), ByVal allAddedRows As Global.System.Collections.Generic.List(Of Global.System.Data.DataRow)) As Integer

Dim result As Integer = 0

If (Not (Me.\_adminTableAdapter) Is Nothing) Then

Dim updatedRows() As Global.System.Data.DataRow = dataSet.admin.Select(Nothing, Nothing, Global.System.Data.DataViewRowState.ModifiedCurrent)

updatedRows = Me.GetRealUpdatedRows(updatedRows, allAddedRows)

If ((Not (updatedRows) Is Nothing) \_

AndAlso (0 < updatedRows.Length)) Then

result = (result + Me.\_adminTableAdapter.Update(updatedRows))

allChangedRows.AddRange(updatedRows)

End If

End If

If (Not (Me.\_stockTableAdapter) Is Nothing) Then

Dim updatedRows() As Global.System.Data.DataRow = dataSet.stock.Select(Nothing, Nothing, Global.System.Data.DataViewRowState.ModifiedCurrent)

updatedRows = Me.GetRealUpdatedRows(updatedRows, allAddedRows)

If ((Not (updatedRows) Is Nothing) \_

AndAlso (0 < updatedRows.Length)) Then

result = (result + Me.\_stockTableAdapter.Update(updatedRows))

allChangedRows.AddRange(updatedRows)

End If

End If

Return result

End Function

'''<summary>

'''Insert rows in top-down order.

'''</summary>

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Function UpdateInsertedRows(ByVal dataSet As InvoiceDataSet, ByVal allAddedRows As Global.System.Collections.Generic.List(Of Global.System.Data.DataRow)) As Integer

Dim result As Integer = 0

If (Not (Me.\_adminTableAdapter) Is Nothing) Then

Dim addedRows() As Global.System.Data.DataRow = dataSet.admin.Select(Nothing, Nothing, Global.System.Data.DataViewRowState.Added)

If ((Not (addedRows) Is Nothing) \_

AndAlso (0 < addedRows.Length)) Then

result = (result + Me.\_adminTableAdapter.Update(addedRows))

allAddedRows.AddRange(addedRows)

End If

End If

If (Not (Me.\_stockTableAdapter) Is Nothing) Then

Dim addedRows() As Global.System.Data.DataRow = dataSet.stock.Select(Nothing, Nothing, Global.System.Data.DataViewRowState.Added)

If ((Not (addedRows) Is Nothing) \_

AndAlso (0 < addedRows.Length)) Then

result = (result + Me.\_stockTableAdapter.Update(addedRows))

allAddedRows.AddRange(addedRows)

End If

End If

Return result

End Function

'''<summary>

'''Delete rows in bottom-up order.

'''</summary>

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Function UpdateDeletedRows(ByVal dataSet As InvoiceDataSet, ByVal allChangedRows As Global.System.Collections.Generic.List(Of Global.System.Data.DataRow)) As Integer

Dim result As Integer = 0

If (Not (Me.\_stockTableAdapter) Is Nothing) Then

Dim deletedRows() As Global.System.Data.DataRow = dataSet.stock.Select(Nothing, Nothing, Global.System.Data.DataViewRowState.Deleted)

If ((Not (deletedRows) Is Nothing) \_

AndAlso (0 < deletedRows.Length)) Then

result = (result + Me.\_stockTableAdapter.Update(deletedRows))

allChangedRows.AddRange(deletedRows)

End If

End If

If (Not (Me.\_adminTableAdapter) Is Nothing) Then

Dim deletedRows() As Global.System.Data.DataRow = dataSet.admin.Select(Nothing, Nothing, Global.System.Data.DataViewRowState.Deleted)

If ((Not (deletedRows) Is Nothing) \_

AndAlso (0 < deletedRows.Length)) Then

result = (result + Me.\_adminTableAdapter.Update(deletedRows))

allChangedRows.AddRange(deletedRows)

End If

End If

Return result

End Function

'''<summary>

'''Remove inserted rows that become updated rows after calling TableAdapter.Update(inserted rows) first

'''</summary>

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Function GetRealUpdatedRows(ByVal updatedRows() As Global.System.Data.DataRow, ByVal allAddedRows As Global.System.Collections.Generic.List(Of Global.System.Data.DataRow)) As Global.System.Data.DataRow()

If ((updatedRows Is Nothing) \_

OrElse (updatedRows.Length < 1)) Then

Return updatedRows

End If

If ((allAddedRows Is Nothing) \_

OrElse (allAddedRows.Count < 1)) Then

Return updatedRows

End If

Dim realUpdatedRows As Global.System.Collections.Generic.List(Of Global.System.Data.DataRow) = New Global.System.Collections.Generic.List(Of Global.System.Data.DataRow)()

Dim i As Integer = 0

Do While (i < updatedRows.Length)

Dim row As Global.System.Data.DataRow = updatedRows(i)

If (allAddedRows.Contains(row) = false) Then

realUpdatedRows.Add(row)

End If

i = (i + 1)

Loop

Return realUpdatedRows.ToArray

End Function

'''<summary>

'''Update all changes to the dataset.

'''</summary>

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Overridable Function UpdateAll(ByVal dataSet As InvoiceDataSet) As Integer

If (dataSet Is Nothing) Then

Throw New Global.System.ArgumentNullException("dataSet")

End If

If (dataSet.HasChanges = false) Then

Return 0

End If

If ((Not (Me.\_adminTableAdapter) Is Nothing) \_

AndAlso (Me.MatchTableAdapterConnection(Me.\_adminTableAdapter.Connection) = false)) Then

Throw New Global.System.ArgumentException("All TableAdapters managed by a TableAdapterManager must use the same connection s"& \_

"tring.")

End If

If ((Not (Me.\_stockTableAdapter) Is Nothing) \_

AndAlso (Me.MatchTableAdapterConnection(Me.\_stockTableAdapter.Connection) = false)) Then

Throw New Global.System.ArgumentException("All TableAdapters managed by a TableAdapterManager must use the same connection s"& \_

"tring.")

End If

Dim workConnection As Global.System.Data.IDbConnection = Me.Connection

If (workConnection Is Nothing) Then

Throw New Global.System.ApplicationException("TableAdapterManager contains no connection information. Set each TableAdapterMana"& \_

"ger TableAdapter property to a valid TableAdapter instance.")

End If

Dim workConnOpened As Boolean = false

If ((workConnection.State And Global.System.Data.ConnectionState.Broken) \_

= Global.System.Data.ConnectionState.Broken) Then

workConnection.Close

End If

If (workConnection.State = Global.System.Data.ConnectionState.Closed) Then

workConnection.Open

workConnOpened = true

End If

Dim workTransaction As Global.System.Data.IDbTransaction = workConnection.BeginTransaction

If (workTransaction Is Nothing) Then

Throw New Global.System.ApplicationException("The transaction cannot begin. The current data connection does not support transa"& \_

"ctions or the current state is not allowing the transaction to begin.")

End If

Dim allChangedRows As Global.System.Collections.Generic.List(Of Global.System.Data.DataRow) = New Global.System.Collections.Generic.List(Of Global.System.Data.DataRow)()

Dim allAddedRows As Global.System.Collections.Generic.List(Of Global.System.Data.DataRow) = New Global.System.Collections.Generic.List(Of Global.System.Data.DataRow)()

Dim adaptersWithAcceptChangesDuringUpdate As Global.System.Collections.Generic.List(Of Global.System.Data.Common.DataAdapter) = New Global.System.Collections.Generic.List(Of Global.System.Data.Common.DataAdapter)()

Dim revertConnections As Global.System.Collections.Generic.Dictionary(Of Object, Global.System.Data.IDbConnection) = New Global.System.Collections.Generic.Dictionary(Of Object, Global.System.Data.IDbConnection)()

Dim result As Integer = 0

Dim backupDataSet As Global.System.Data.DataSet = Nothing

If Me.BackupDataSetBeforeUpdate Then

backupDataSet = New Global.System.Data.DataSet()

backupDataSet.Merge(dataSet)

End If

Try

'---- Prepare for update -----------

'

If (Not (Me.\_adminTableAdapter) Is Nothing) Then

revertConnections.Add(Me.\_adminTableAdapter, Me.\_adminTableAdapter.Connection)

Me.\_adminTableAdapter.Connection = CType(workConnection,Global.System.Data.OleDb.OleDbConnection)

Me.\_adminTableAdapter.Transaction = CType(workTransaction,Global.System.Data.OleDb.OleDbTransaction)

If Me.\_adminTableAdapter.Adapter.AcceptChangesDuringUpdate Then

Me.\_adminTableAdapter.Adapter.AcceptChangesDuringUpdate = false

adaptersWithAcceptChangesDuringUpdate.Add(Me.\_adminTableAdapter.Adapter)

End If

End If

If (Not (Me.\_stockTableAdapter) Is Nothing) Then

revertConnections.Add(Me.\_stockTableAdapter, Me.\_stockTableAdapter.Connection)

Me.\_stockTableAdapter.Connection = CType(workConnection,Global.System.Data.OleDb.OleDbConnection)

Me.\_stockTableAdapter.Transaction = CType(workTransaction,Global.System.Data.OleDb.OleDbTransaction)

If Me.\_stockTableAdapter.Adapter.AcceptChangesDuringUpdate Then

Me.\_stockTableAdapter.Adapter.AcceptChangesDuringUpdate = false

adaptersWithAcceptChangesDuringUpdate.Add(Me.\_stockTableAdapter.Adapter)

End If

End If

'

'---- Perform updates -----------

'

If (Me.UpdateOrder = UpdateOrderOption.UpdateInsertDelete) Then

result = (result + Me.UpdateUpdatedRows(dataSet, allChangedRows, allAddedRows))

result = (result + Me.UpdateInsertedRows(dataSet, allAddedRows))

Else

result = (result + Me.UpdateInsertedRows(dataSet, allAddedRows))

result = (result + Me.UpdateUpdatedRows(dataSet, allChangedRows, allAddedRows))

End If

result = (result + Me.UpdateDeletedRows(dataSet, allChangedRows))

'

'---- Commit updates -----------

'

workTransaction.Commit

If (0 < allAddedRows.Count) Then

Dim rows((allAddedRows.Count) - 1) As Global.System.Data.DataRow

allAddedRows.CopyTo(rows)

Dim i As Integer = 0

Do While (i < rows.Length)

Dim row As Global.System.Data.DataRow = rows(i)

row.AcceptChanges

i = (i + 1)

Loop

End If

If (0 < allChangedRows.Count) Then

Dim rows((allChangedRows.Count) - 1) As Global.System.Data.DataRow

allChangedRows.CopyTo(rows)

Dim i As Integer = 0

Do While (i < rows.Length)

Dim row As Global.System.Data.DataRow = rows(i)

row.AcceptChanges

i = (i + 1)

Loop

End If

Catch ex As Global.System.Exception

workTransaction.Rollback

'---- Restore the dataset -----------

If Me.BackupDataSetBeforeUpdate Then

Global.System.Diagnostics.Debug.Assert((Not (backupDataSet) Is Nothing))

dataSet.Clear

dataSet.Merge(backupDataSet)

Else

If (0 < allAddedRows.Count) Then

Dim rows((allAddedRows.Count) - 1) As Global.System.Data.DataRow

allAddedRows.CopyTo(rows)

Dim i As Integer = 0

Do While (i < rows.Length)

Dim row As Global.System.Data.DataRow = rows(i)

row.AcceptChanges

row.SetAdded

i = (i + 1)

Loop

End If

End If

Throw ex

Finally

If workConnOpened Then

workConnection.Close

End If

If (Not (Me.\_adminTableAdapter) Is Nothing) Then

Me.\_adminTableAdapter.Connection = CType(revertConnections(Me.\_adminTableAdapter),Global.System.Data.OleDb.OleDbConnection)

Me.\_adminTableAdapter.Transaction = Nothing

End If

If (Not (Me.\_stockTableAdapter) Is Nothing) Then

Me.\_stockTableAdapter.Connection = CType(revertConnections(Me.\_stockTableAdapter),Global.System.Data.OleDb.OleDbConnection)

Me.\_stockTableAdapter.Transaction = Nothing

End If

If (0 < adaptersWithAcceptChangesDuringUpdate.Count) Then

Dim adapters((adaptersWithAcceptChangesDuringUpdate.Count) - 1) As Global.System.Data.Common.DataAdapter

adaptersWithAcceptChangesDuringUpdate.CopyTo(adapters)

Dim i As Integer = 0

Do While (i < adapters.Length)

Dim adapter As Global.System.Data.Common.DataAdapter = adapters(i)

adapter.AcceptChangesDuringUpdate = true

i = (i + 1)

Loop

End If

End Try

Return result

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overridable Sub SortSelfReferenceRows(ByVal rows() As Global.System.Data.DataRow, ByVal relation As Global.System.Data.DataRelation, ByVal childFirst As Boolean)

Global.System.Array.Sort(Of Global.System.Data.DataRow)(rows, New SelfReferenceComparer(relation, childFirst))

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Protected Overridable Function MatchTableAdapterConnection(ByVal inputConnection As Global.System.Data.IDbConnection) As Boolean

If (Not (Me.\_connection) Is Nothing) Then

Return true

End If

If ((Me.Connection Is Nothing) \_

OrElse (inputConnection Is Nothing)) Then

Return true

End If

If String.Equals(Me.Connection.ConnectionString, inputConnection.ConnectionString, Global.System.StringComparison.Ordinal) Then

Return true

End If

Return false

End Function

'''<summary>

'''Update Order Option

'''</summary>

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Enum UpdateOrderOption

InsertUpdateDelete = 0

UpdateInsertDelete = 1

End Enum

'''<summary>

'''Used to sort self-referenced table's rows

'''</summary>

<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Class SelfReferenceComparer

Inherits Object

Implements Global.System.Collections.Generic.IComparer(Of Global.System.Data.DataRow)

Private \_relation As Global.System.Data.DataRelation

Private \_childFirst As Integer

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Friend Sub New(ByVal relation As Global.System.Data.DataRelation, ByVal childFirst As Boolean)

MyBase.New

Me.\_relation = relation

If childFirst Then

Me.\_childFirst = -1

Else

Me.\_childFirst = 1

End If

End Sub

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Private Function GetRoot(ByVal row As Global.System.Data.DataRow, ByRef distance As Integer) As Global.System.Data.DataRow

Global.System.Diagnostics.Debug.Assert((Not (row) Is Nothing))

Dim root As Global.System.Data.DataRow = row

distance = 0

Dim traversedRows As Global.System.Collections.Generic.IDictionary(Of Global.System.Data.DataRow, Global.System.Data.DataRow) = New Global.System.Collections.Generic.Dictionary(Of Global.System.Data.DataRow, Global.System.Data.DataRow)()

traversedRows(row) = row

Dim parent As Global.System.Data.DataRow = row.GetParentRow(Me.\_relation, Global.System.Data.DataRowVersion.[Default])

Do While ((Not (parent) Is Nothing) \_

AndAlso (traversedRows.ContainsKey(parent) = false))

distance = (distance + 1)

root = parent

traversedRows(parent) = parent

parent = parent.GetParentRow(Me.\_relation, Global.System.Data.DataRowVersion.[Default])

Loop

If (distance = 0) Then

traversedRows.Clear

traversedRows(row) = row

parent = row.GetParentRow(Me.\_relation, Global.System.Data.DataRowVersion.Original)

Do While ((Not (parent) Is Nothing) \_

AndAlso (traversedRows.ContainsKey(parent) = false))

distance = (distance + 1)

root = parent

traversedRows(parent) = parent

parent = parent.GetParentRow(Me.\_relation, Global.System.Data.DataRowVersion.Original)

Loop

End If

Return root

End Function

<Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), \_

Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")> \_

Public Function Compare(ByVal row1 As Global.System.Data.DataRow, ByVal row2 As Global.System.Data.DataRow) As Integer Implements Global.System.Collections.Generic.IComparer(Of Global.System.Data.DataRow).Compare

If Object.ReferenceEquals(row1, row2) Then

Return 0

End If

If (row1 Is Nothing) Then

Return -1

End If

If (row2 Is Nothing) Then

Return 1

End If

Dim distance1 As Integer = 0

Dim root1 As Global.System.Data.DataRow = Me.GetRoot(row1, distance1)

Dim distance2 As Integer = 0

Dim root2 As Global.System.Data.DataRow = Me.GetRoot(row2, distance2)

If Object.ReferenceEquals(root1, root2) Then

Return (Me.\_childFirst \* distance1.CompareTo(distance2))

Else

Global.System.Diagnostics.Debug.Assert(((Not (root1.Table) Is Nothing) \_

AndAlso (Not (root2.Table) Is Nothing)))

If (root1.Table.Rows.IndexOf(root1) < root2.Table.Rows.IndexOf(root2)) Then

Return -1

Else

Return 1

End If

End If

End Function

End Class

End Class

End Namespace