www.bytescout.com

How to generate report from SQL server VB in Crystal Reports and ByteScout Barcode SDK

Write code in Crystal Reports to generate report from SQL server VB with this step-by-step tutorial

On this page you will learn from code samples for programming in Crystal Reports.Writing of the code to generate report from SQL server VB in Crystal Reports can be done by developers of any level using ByteScout Barcode SDK. ByteScout Barcode SDK is the fully featured library to generate barcodes. Supports QR Code, Code 39, Code 128, UPC, GS1, GS-128, PDF417, Datamatrix and many other barcode types. Includes various options for barcode generation to ensure output quality, add barcodes to new or existing pdf files and images and you can use it to generate report from SQL server VB with Crystal Reports.

The SDK samples like this one below explain how to quickly make your application do generate report from SQL server VB in Crystal Reports with the help of ByteScout Barcode SDK. This Crystal Reports sample code is all you need for your app. Just copy and paste the code, add references (if needs to) and you are all set! Test Crystal Reports sample code examples whether they respond your needs and requirements for the project.

You can download free trial version of ByteScout Barcode SDK from our website to see and try many others source code samples for Crystal Reports.

FOR MORE INFORMATION AND FREE TRIAL:

Download Free Trial SDK (on-premise version)

Read more about ByteScout Barcode SDK

Explore API Documentation

Get Free Training for ByteScout Barcode SDK

Get Free API key for Web API

visit www.ByteScout.com

Source Code Files:

CrystalReport1.vb

```
'------------------------------------------------------------------------------
' <auto-generated>
'     This code was generated by a tool.
'     Runtime Version:2.0.50727.4927
'
'     Changes to this file may cause incorrect behavior and will be lost if
'     the code is regenerated.
' </auto-generated>
'------------------------------------------------------------------------------

Imports System.ComponentModel
Imports CrystalDecisions.Shared
Imports CrystalDecisions.ReportSource
Imports CrystalDecisions.CrystalReports.Engine


Public Class CrystalReport1
        Inherits ReportClass

        Public Sub New()
        End Sub

        Public Overloads Overrides Property ResourceName() As String
                Get
                        Return "CrystalReport1.rpt"
                End Get
                                ' Do nothing
                Set
                End Set
        End Property

        <Browsable(False)> _
        <DesignerSerializationVisibilityAttribute(System.ComponentModel.DesignerSerial
        Public ReadOnly Property Section1() As CrystalDecisions.CrystalReports.Engine.S
                Get
                        Return Me.ReportDefinition.Sections(0)
                End Get
        End Property

        <Browsable(False)> _
        <DesignerSerializationVisibilityAttribute(System.ComponentModel.DesignerSerial
        Public ReadOnly Property Section2() As CrystalDecisions.CrystalReports.Engine.S
                Get
                        Return Me.ReportDefinition.Sections(1)
                End Get
        End Property

        <Browsable(False)> _
        <DesignerSerializationVisibilityAttribute(System.ComponentModel.DesignerSerial
        Public ReadOnly Property Section3() As CrystalDecisions.CrystalReports.Engine.S
                Get
                        Return Me.ReportDefinition.Sections(2)
```

```
                End Get
        End Property

        <Browsable(False)> _
        <DesignerSerializationVisibilityAttribute(System.ComponentModel.DesignerSerial
        Public ReadOnly Property Section4() As CrystalDecisions.CrystalReports.Engine.S
                Get
                        Return Me.ReportDefinition.Sections(3)
                End Get
        End Property

        <Browsable(False)> _
        <DesignerSerializationVisibilityAttribute(System.ComponentModel.DesignerSerial
        Public ReadOnly Property Section5() As CrystalDecisions.CrystalReports.Engine.S
                Get
                        Return Me.ReportDefinition.Sections(4)
                End Get
        End Property
End Class

<System.Drawing.ToolboxBitmapAttribute(GetType(CrystalDecisions.Shared.ExportOptions),
Public Class CachedCrystalReport1
        Inherits Component
        Implements ICachedReport

        Public Sub New()
        End Sub

        <Browsable(False)> _
        <DesignerSerializationVisibilityAttribute(System.ComponentModel.DesignerSerial
        Public Overridable Property IsCacheable() As Boolean Implements ICachedReport.I
                Get
                        Return True
                End Get
                                '
                Set
                End Set
        End Property

        <Browsable(False)> _
        <DesignerSerializationVisibilityAttribute(System.ComponentModel.DesignerSerial
        Public Overridable Property ShareDBLogonInfo() As Boolean Implements ICachedRep
                Get
                        Return False
                End Get
                                '
                Set
                End Set
        End Property

        <Browsable(False)> _
        <DesignerSerializationVisibilityAttribute(System.ComponentModel.DesignerSerial
        Public Overridable Property CacheTimeOut() As System.TimeSpan Implements ICache
                Get
                        Return CachedReportConstants.DEFAULT_TIMEOUT
                End Get
                                '
                Set
                End Set
        End Property
```

```
        Public Overridable Function CreateReport() As CrystalDecisions.CrystalReports.
                Dim rpt As New CrystalReport1()
                rpt.Site = Me.Site
                Return rpt
        End Function

        Public Overridable Function GetCustomizedCacheKey(request As RequestContext) As
                Dim key As [String] = Nothing
                ' // The following is the code used to generate the default
                ' // cache key for caching report jobs in the ASP.NET Cache.
                ' // Feel free to modify this code to suit your needs.
                ' // Returning key == null causes the default cache key to
                ' // be generated.
                '
                ' key = RequestContext.BuildCompleteCacheKey(
                '     request,
                '     null,       // sReportFilename
                '     this.GetType(),
                '     this.ShareDBLogonInfo );
                Return key
        End Function
End Class
```

DataSet1.Designer.vb

```
'------------------------------------------------------------------------------
' <auto-generated>
'     This code was generated by a tool.
'     Runtime Version:2.0.50727.4927
'
'     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.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSe
 Global.System.Serializable(),  _
 Global.System.ComponentModel.DesignerCategoryAttribute("code"),  _
 Global.System.ComponentModel.ToolboxItem(true),  _
 Global.System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedDataSetSchema"),
 Global.System.Xml.Serialization.XmlRootAttribute("DataSet1"),  _
```

```
 Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.DataSet")>  _
Partial Public Class DataSet1
    Inherits Global.System.Data.DataSet

    Private tableProducts As ProductsDataTable

    Private _schemaSerializationMode As Global.System.Data.SchemaSerializationMode = G

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Sub New()
        MyBase.New
        Me.BeginInit
        Me.InitClass
        Dim schemaChangedHandler As Global.System.ComponentModel.CollectionChangeEventH
        AddHandler MyBase.Tables.CollectionChanged, schemaChangedHandler
        AddHandler MyBase.Relations.CollectionChanged, schemaChangedHandler
        Me.EndInit
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Protected Sub New(ByVal info As Global.System.Runtime.Serialization.SerializationIn
        MyBase.New(info, context, false)
        If (Me.IsBinarySerialized(info, context) = true) Then
            Me.InitVars(false)
            Dim schemaChangedHandler1 As Global.System.ComponentModel.CollectionChangeE
            AddHandler Me.Tables.CollectionChanged, schemaChangedHandler1
            AddHandler Me.Relations.CollectionChanged, schemaChangedHandler1
            Return
        End If
        Dim strSchema As String = CType(info.GetValue("XmlSchema", GetType(String)),Str
        If (Me.DetermineSchemaSerializationMode(info, context) = Global.System.Data.Sch
            Dim ds As Global.System.Data.DataSet = New Global.System.Data.DataSet
            ds.ReadXmlSchema(New Global.System.Xml.XmlTextReader(New Global.System.IO.S
            If (Not (ds.Tables("Products")) Is Nothing) Then
                MyBase.Tables.Add(New ProductsDataTable(ds.Tables("Products")))
            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.S
        End If
        Me.GetSerializationData(info, context)
        Dim schemaChangedHandler As Global.System.ComponentModel.CollectionChangeEventH
        AddHandler MyBase.Tables.CollectionChanged, schemaChangedHandler
        AddHandler Me.Relations.CollectionChanged, schemaChangedHandler
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
     Global.System.ComponentModel.Browsable(false),  _
     Global.System.ComponentModel.DesignerSerializationVisibility(Global.System.Compone
    Public ReadOnly Property Products() As ProductsDataTable
        Get
            Return Me.tableProducts
        End Get
```

```
    End Property

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
     Global.System.ComponentModel.BrowsableAttribute(true),  _
     Global.System.ComponentModel.DesignerSerializationVisibilityAttribute(Global.Syste
    Public Overrides Property SchemaSerializationMode() As Global.System.Data.SchemaSer
        Get
            Return Me._schemaSerializationMode
        End Get
        Set
            Me._schemaSerializationMode = value
        End Set
    End Property

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
     Global.System.ComponentModel.DesignerSerializationVisibilityAttribute(Global.Syste
    Public Shadows ReadOnly Property Tables() As Global.System.Data.DataTableCollectio
        Get
            Return MyBase.Tables
        End Get
    End Property

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
     Global.System.ComponentModel.DesignerSerializationVisibilityAttribute(Global.Syste
    Public Shadows ReadOnly Property Relations() As Global.System.Data.DataRelationCol
        Get
            Return MyBase.Relations
        End Get
    End Property

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Protected Overrides Sub InitializeDerivedDataSet()
        Me.BeginInit
        Me.InitClass
        Me.EndInit
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Overrides Function Clone() As Global.System.Data.DataSet
        Dim cln As DataSet1 = CType(MyBase.Clone,DataSet1)
        cln.InitVars
        cln.SchemaSerializationMode = Me.SchemaSerializationMode
        Return cln
    End Function

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Protected Overrides Function ShouldSerializeTables() As Boolean
        Return false
    End Function

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Protected Overrides Function ShouldSerializeRelations() As Boolean
        Return false
    End Function

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Protected Overrides Sub ReadXmlSerializable(ByVal reader As Global.System.Xml.XmlRe
        If (Me.DetermineSchemaSerializationMode(reader) = Global.System.Data.SchemaSer
            Me.Reset
            Dim ds As Global.System.Data.DataSet = New Global.System.Data.DataSet
```

```
            ds.ReadXml(reader)
            If (Not (ds.Tables("Products")) Is Nothing) Then
                MyBase.Tables.Add(New ProductsDataTable(ds.Tables("Products")))
            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()>  _
    Protected Overrides Function GetSchemaSerializable() As Global.System.Xml.Schema.Xm
        Dim stream As Global.System.IO.MemoryStream = New Global.System.IO.MemoryStrean
        Me.WriteXmlSchema(New Global.System.Xml.XmlTextWriter(stream, Nothing))
        stream.Position = 0
        Return Global.System.Xml.Schema.XmlSchema.Read(New Global.System.Xml.XmlTextRe
    End Function

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Friend Overloads Sub InitVars()
        Me.InitVars(true)
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Friend Overloads Sub InitVars(ByVal initTable As Boolean)
        Me.tableProducts = CType(MyBase.Tables("Products"),ProductsDataTable)
        If (initTable = true) Then
            If (Not (Me.tableProducts) Is Nothing) Then
                Me.tableProducts.InitVars
            End If
        End If
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Private Sub InitClass()
        Me.DataSetName = "DataSet1"
        Me.Prefix = ""
        Me.Namespace = "http://tempuri.org/DataSet1.xsd"
        Me.EnforceConstraints = true
        Me.SchemaSerializationMode = Global.System.Data.SchemaSerializationMode.Include
        Me.tableProducts = New ProductsDataTable
        MyBase.Tables.Add(Me.tableProducts)
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Private Function ShouldSerializeProducts() As Boolean
        Return false
    End Function

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Private Sub SchemaChanged(ByVal sender As Object, ByVal e As Global.System.Componer
        If (e.Action = Global.System.ComponentModel.CollectionChangeAction.Remove) Ther
```

```
            Me.InitVars
        End If
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Shared Function GetTypedDataSetSchema(ByVal xs As Global.System.Xml.Schema.X
        Dim ds As DataSet1 = New DataSet1
        Dim type As Global.System.Xml.Schema.XmlSchemaComplexType = New Global.System.X
        Dim sequence As Global.System.Xml.Schema.XmlSchemaSequence = New Global.System
        Dim any As Global.System.Xml.Schema.XmlSchemaAny = New Global.System.Xml.Schema
        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.MemoryStrea
            Dim s2 As Global.System.IO.MemoryStream = New Global.System.IO.MemoryStrea
            Try
                Dim schema As Global.System.Xml.Schema.XmlSchema = Nothing
                dsSchema.Write(s1)
                Dim schemas As Global.System.Collections.IEnumerator = xs.Schemas(dsSc
                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

    Public Delegate Sub ProductsRowChangeEventHandler(ByVal sender As Object, ByVal e A

    '''<summary>
    '''Represents the strongly named DataTable class.
    '''</summary>
    <Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDa
     Global.System.Serializable(),  _
```

```
    Global.System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedTableSchema")>
    Partial Public Class ProductsDataTable
        Inherits Global.System.Data.DataTable
        Implements Global.System.Collections.IEnumerable

        Private columnProduct_ID As Global.System.Data.DataColumn

        Private columnProduct_Name As Global.System.Data.DataColumn

        Private columnProduct_Description As Global.System.Data.DataColumn

        Private columnBarCode As Global.System.Data.DataColumn

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Sub New()
            MyBase.New
            Me.TableName = "Products"
            Me.BeginInit
            Me.InitClass
            Me.EndInit
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        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()>  _
        Protected Sub New(ByVal info As Global.System.Runtime.Serialization.Serializat
            MyBase.New(info, context)
            Me.InitVars
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public ReadOnly Property Product_IDColumn() As Global.System.Data.DataColumn
            Get
                Return Me.columnProduct_ID
            End Get
        End Property

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public ReadOnly Property Product_NameColumn() As Global.System.Data.DataColumn
            Get
                Return Me.columnProduct_Name
            End Get
        End Property

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
```

```
        Public ReadOnly Property Product_DescriptionColumn() As Global.System.Data.Data
            Get
                Return Me.columnProduct_Description
            End Get
        End Property

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public ReadOnly Property BarCodeColumn() As Global.System.Data.DataColumn
            Get
                Return Me.columnBarCode
            End Get
        End Property

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
         Global.System.ComponentModel.Browsable(false)>  _
        Public ReadOnly Property Count() As Integer
            Get
                Return Me.Rows.Count
            End Get
        End Property

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Default ReadOnly Property Item(ByVal index As Integer) As ProductsRow
            Get
                Return CType(Me.Rows(index),ProductsRow)
            End Get
        End Property

        Public Event ProductsRowChanging As ProductsRowChangeEventHandler

        Public Event ProductsRowChanged As ProductsRowChangeEventHandler

        Public Event ProductsRowDeleting As ProductsRowChangeEventHandler

        Public Event ProductsRowDeleted As ProductsRowChangeEventHandler

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Overloads Sub AddProductsRow(ByVal row As ProductsRow)
            Me.Rows.Add(row)
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Overloads Function AddProductsRow(ByVal Product_ID As Integer, ByVal Pro
            Dim rowProductsRow As ProductsRow = CType(Me.NewRow,ProductsRow)
            Dim columnValuesArray() As Object = New Object() {Product_ID, Product_Name
            rowProductsRow.ItemArray = columnValuesArray
            Me.Rows.Add(rowProductsRow)
            Return rowProductsRow
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Overridable Function GetEnumerator() As Global.System.Collections.IEnume
            Return Me.Rows.GetEnumerator
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Overrides Function Clone() As Global.System.Data.DataTable
            Dim cln As ProductsDataTable = CType(MyBase.Clone,ProductsDataTable)
            cln.InitVars
            Return cln
```

```
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Protected Overrides Function CreateInstance() As Global.System.Data.DataTable
            Return New ProductsDataTable
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Friend Sub InitVars()
            Me.columnProduct_ID = MyBase.Columns("Product ID")
            Me.columnProduct_Name = MyBase.Columns("Product Name")
            Me.columnProduct_Description = MyBase.Columns("Product Description")
            Me.columnBarCode = MyBase.Columns("BarCode")
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Private Sub InitClass()
            Me.columnProduct_ID = New Global.System.Data.DataColumn("Product ID", GetTy
            MyBase.Columns.Add(Me.columnProduct_ID)
            Me.columnProduct_Name = New Global.System.Data.DataColumn("Product Name", C
            MyBase.Columns.Add(Me.columnProduct_Name)
            Me.columnProduct_Description = New Global.System.Data.DataColumn("Product D
            MyBase.Columns.Add(Me.columnProduct_Description)
            Me.columnBarCode = New Global.System.Data.DataColumn("BarCode", GetType(Byt
            MyBase.Columns.Add(Me.columnBarCode)
            Me.columnProduct_Name.MaxLength = 100
            Me.columnProduct_Description.MaxLength = 255
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Function NewProductsRow() As ProductsRow
            Return CType(Me.NewRow,ProductsRow)
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Protected Overrides Function NewRowFromBuilder(ByVal builder As Global.System.D
            Return New ProductsRow(builder)
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Protected Overrides Function GetRowType() As Global.System.Type
            Return GetType(ProductsRow)
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Protected Overrides Sub OnRowChanged(ByVal e As Global.System.Data.DataRowChang
            MyBase.OnRowChanged(e)
            If (Not (Me.ProductsRowChangedEvent) Is Nothing) Then
                RaiseEvent ProductsRowChanged(Me, New ProductsRowChangeEvent(CType(e.Ro
            End If
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Protected Overrides Sub OnRowChanging(ByVal e As Global.System.Data.DataRowChan
            MyBase.OnRowChanging(e)
            If (Not (Me.ProductsRowChangingEvent) Is Nothing) Then
                RaiseEvent ProductsRowChanging(Me, New ProductsRowChangeEvent(CType(e.R
            End If
        End Sub
```

```
        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Protected Overrides Sub OnRowDeleted(ByVal e As Global.System.Data.DataRowChang
            MyBase.OnRowDeleted(e)
            If (Not (Me.ProductsRowDeletedEvent) Is Nothing) Then
                RaiseEvent ProductsRowDeleted(Me, New ProductsRowChangeEvent(CType(e.Ro
            End If
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Protected Overrides Sub OnRowDeleting(ByVal e As Global.System.Data.DataRowChan
            MyBase.OnRowDeleting(e)
            If (Not (Me.ProductsRowDeletingEvent) Is Nothing) Then
                RaiseEvent ProductsRowDeleting(Me, New ProductsRowChangeEvent(CType(e.R
            End If
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Sub RemoveProductsRow(ByVal row As ProductsRow)
            Me.Rows.Remove(row)
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Shared Function GetTypedTableSchema(ByVal xs As Global.System.Xml.Schema
            Dim type As Global.System.Xml.Schema.XmlSchemaComplexType = New Global.Syst
            Dim sequence As Global.System.Xml.Schema.XmlSchemaSequence = New Global.Sys
            Dim ds As DataSet1 = New DataSet1
            Dim any1 As Global.System.Xml.Schema.XmlSchemaAny = New Global.System.Xml.S
            any1.Namespace = "http://www.w3.org/2001/XMLSchema"
            any1.MinOccurs = New Decimal(0)
            any1.MaxOccurs = Decimal.MaxValue
            any1.ProcessContents = Global.System.Xml.Schema.XmlSchemaContentProcessing
            sequence.Items.Add(any1)
            Dim any2 As Global.System.Xml.Schema.XmlSchemaAny = New Global.System.Xml.S
            any2.Namespace = "urn:schemas-microsoft-com:xml-diffgram-v1"
            any2.MinOccurs = New Decimal(1)
            any2.ProcessContents = Global.System.Xml.Schema.XmlSchemaContentProcessing
            sequence.Items.Add(any2)
            Dim attribute1 As Global.System.Xml.Schema.XmlSchemaAttribute = New Global
            attribute1.Name = "namespace"
            attribute1.FixedValue = ds.Namespace
            type.Attributes.Add(attribute1)
            Dim attribute2 As Global.System.Xml.Schema.XmlSchemaAttribute = New Global
            attribute2.Name = "tableTypeName"
            attribute2.FixedValue = "ProductsDataTable"
            type.Attributes.Add(attribute2)
            type.Particle = sequence
            Dim dsSchema As Global.System.Xml.Schema.XmlSchema = ds.GetSchemaSerializab
            If xs.Contains(dsSchema.TargetNamespace) Then
                Dim s1 As Global.System.IO.MemoryStream = New Global.System.IO.MemorySt
                Dim s2 As Global.System.IO.MemoryStream = New Global.System.IO.MemorySt
                Try
                    Dim schema As Global.System.Xml.Schema.XmlSchema = Nothing
                    dsSchema.Write(s1)
                    Dim schemas As Global.System.Collections.IEnumerator = xs.Schemas(d
                    Do While schemas.MoveNext
                        schema = CType(schemas.Current,Global.System.Xml.Schema.XmlSche
                        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>
<Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDa
Partial Public Class ProductsRow
    Inherits Global.System.Data.DataRow

    Private tableProducts As ProductsDataTable

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Friend Sub New(ByVal rb As Global.System.Data.DataRowBuilder)
        MyBase.New(rb)
        Me.tableProducts = CType(Me.Table,ProductsDataTable)
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Property Product_ID() As Integer
        Get
            Try
                Return CType(Me(Me.tableProducts.Product_IDColumn),Integer)
            Catch e As Global.System.InvalidCastException
                Throw New Global.System.Data.StrongTypingException("The value for
            End Try
        End Get
        Set
            Me(Me.tableProducts.Product_IDColumn) = value
        End Set
    End Property

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Property Product_Name() As String
        Get
            Try
```

```
                Return CType(Me(Me.tableProducts.Product_NameColumn),String)
            Catch e As Global.System.InvalidCastException
                Throw New Global.System.Data.StrongTypingException("The value for c
            End Try
        End Get
        Set
            Me(Me.tableProducts.Product_NameColumn) = value
        End Set
    End Property

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Property Product_Description() As String
        Get
            Try
                Return CType(Me(Me.tableProducts.Product_DescriptionColumn),String)
            Catch e As Global.System.InvalidCastException
                Throw New Global.System.Data.StrongTypingException("The value for c
            End Try
        End Get
        Set
            Me(Me.tableProducts.Product_DescriptionColumn) = value
        End Set
    End Property

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Property BarCode() As Byte()
        Get
            Try
                Return CType(Me(Me.tableProducts.BarCodeColumn),Byte())
            Catch e As Global.System.InvalidCastException
                Throw New Global.System.Data.StrongTypingException("The value for c
            End Try
        End Get
        Set
            Me(Me.tableProducts.BarCodeColumn) = value
        End Set
    End Property

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Function IsProduct_IDNull() As Boolean
        Return Me.IsNull(Me.tableProducts.Product_IDColumn)
    End Function

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Sub SetProduct_IDNull()
        Me(Me.tableProducts.Product_IDColumn) = Global.System.Convert.DBNull
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Function IsProduct_NameNull() As Boolean
        Return Me.IsNull(Me.tableProducts.Product_NameColumn)
    End Function

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Sub SetProduct_NameNull()
        Me(Me.tableProducts.Product_NameColumn) = Global.System.Convert.DBNull
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Function IsProduct_DescriptionNull() As Boolean
```

```
            Return Me.IsNull(Me.tableProducts.Product_DescriptionColumn)
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Sub SetProduct_DescriptionNull()
            Me(Me.tableProducts.Product_DescriptionColumn) = Global.System.Convert.DBNu
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Function IsBarCodeNull() As Boolean
            Return Me.IsNull(Me.tableProducts.BarCodeColumn)
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Sub SetBarCodeNull()
            Me(Me.tableProducts.BarCodeColumn) = Global.System.Convert.DBNull
        End Sub
    End Class

    '''<summary>
    '''Row event argument class
    '''</summary>
    <Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDa
    Public Class ProductsRowChangeEvent
        Inherits Global.System.EventArgs

        Private eventRow As ProductsRow

        Private eventAction As Global.System.Data.DataRowAction

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Sub New(ByVal row As ProductsRow, ByVal action As Global.System.Data.Da
            MyBase.New
            Me.eventRow = row
            Me.eventAction = action
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public ReadOnly Property Row() As ProductsRow
            Get
                Return Me.eventRow
            End Get
        End Property

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public ReadOnly Property Action() As Global.System.Data.DataRowAction
            Get
                Return Me.eventAction
            End Get
        End Property
    End Class
End Class

Namespace DataSet1TableAdapters

    '''<summary>
    '''Represents the connection and commands used to retrieve and save data.
    '''</summary>
    <Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDa
     Global.System.ComponentModel.DesignerCategoryAttribute("code"),  _
```

```
    Global.System.ComponentModel.ToolboxItem(true),  _
    Global.System.ComponentModel.DataObjectAttribute(true),  _
    Global.System.ComponentModel.DesignerAttribute("Microsoft.VSDesigner.DataSource.D
        ", Version=8.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"),  _
    Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter")>
Partial Public Class ProductsTableAdapter
    Inherits Global.System.ComponentModel.Component

    Private WithEvents _adapter As Global.System.Data.SqlClient.SqlDataAdapter

    Private _connection As Global.System.Data.SqlClient.SqlConnection

    Private _commandCollection() As Global.System.Data.SqlClient.SqlCommand

    Private _clearBeforeFill As Boolean

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Public Sub New()
        MyBase.New
        Me.ClearBeforeFill = true
    End Sub

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Private ReadOnly Property Adapter() As Global.System.Data.SqlClient.SqlDataAdap
        Get
            If (Me._adapter Is Nothing) Then
                Me.InitAdapter
            End If
            Return Me._adapter
        End Get
    End Property

    <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
    Friend Property Connection() As Global.System.Data.SqlClient.SqlConnection
        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.SqlClient.Sql
                End If
                i = (i + 1)
            Loop
        End Set
    End Property
```

```
        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Protected ReadOnly Property CommandCollection() As Global.System.Data.SqlClient
            Get
                If (Me._commandCollection Is Nothing) Then
                    Me.InitCommandCollection
                End If
                Return Me._commandCollection
            End Get
        End Property

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Public Property ClearBeforeFill() As Boolean
            Get
                Return Me._clearBeforeFill
            End Get
            Set
                Me._clearBeforeFill = value
            End Set
        End Property

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Private Sub InitAdapter()
            Me._adapter = New Global.System.Data.SqlClient.SqlDataAdapter
            Dim tableMapping As Global.System.Data.Common.DataTableMapping = New Globa
            tableMapping.SourceTable = "Table"
            tableMapping.DataSetTable = "Products"
            tableMapping.ColumnMappings.Add("Product ID", "Product ID")
            tableMapping.ColumnMappings.Add("Product Name", "Product Name")
            tableMapping.ColumnMappings.Add("Product Description", "Product Descriptio
            Me._adapter.TableMappings.Add(tableMapping)
            Me._adapter.InsertCommand = New Global.System.Data.SqlClient.SqlCommand
            Me._adapter.InsertCommand.Connection = Me.Connection
            Me._adapter.InsertCommand.CommandText = "INSERT INTO [dbo].[Products] ([Pr
                ") VALUES (@Product_ID, @Product_Name, @Product_Description)"
            Me._adapter.InsertCommand.CommandType = Global.System.Data.CommandType.Tex
            Me._adapter.InsertCommand.Parameters.Add(New Global.System.Data.SqlClient.
            Me._adapter.InsertCommand.Parameters.Add(New Global.System.Data.SqlClient.
            Me._adapter.InsertCommand.Parameters.Add(New Global.System.Data.SqlClient.
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Private Sub InitConnection()
            Me._connection = New Global.System.Data.SqlClient.SqlConnection
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute()>  _
        Private Sub InitCommandCollection()
            Me._commandCollection = New Global.System.Data.SqlClient.SqlCommand(0) {}
            Me._commandCollection(0) = New Global.System.Data.SqlClient.SqlCommand
            Me._commandCollection(0).Connection = Me.Connection
            Me._commandCollection(0).CommandText = "SELECT [Product ID], [Product Name]
            Me._commandCollection(0).CommandType = Global.System.Data.CommandType.Text
        End Sub

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
         Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapte
         Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.Componen
        Public Overloads Overridable Function Fill(ByVal dataTable As DataSet1.Products
            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.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter
         Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.Componen
        Public Overloads Overridable Function GetData() As DataSet1.ProductsDataTable
            Me.Adapter.SelectCommand = Me.CommandCollection(0)
            Dim dataTable As DataSet1.ProductsDataTable = New DataSet1.ProductsDataTab
            Me.Adapter.Fill(dataTable)
            Return dataTable
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
         Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter
        Public Overloads Overridable Function Update(ByVal dataTable As DataSet1.Produ
            Return Me.Adapter.Update(dataTable)
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
         Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter
        Public Overloads Overridable Function Update(ByVal dataSet As DataSet1) As Inte
            Return Me.Adapter.Update(dataSet, "Products")
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
         Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter
        Public Overloads Overridable Function Update(ByVal dataRow As Global.System.Da
            Return Me.Adapter.Update(New Global.System.Data.DataRow() {dataRow})
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
         Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter
        Public Overloads Overridable Function Update(ByVal dataRows() As Global.System
            Return Me.Adapter.Update(dataRows)
        End Function

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
         Global.System.ComponentModel.Design.HelpKeywordAttribute("vs.data.TableAdapter
         Global.System.ComponentModel.DataObjectMethodAttribute(Global.System.Componen
        Public Overloads Overridable Function Insert(ByVal Product_ID As Global.System
            If (Product_ID.HasValue = true) Then
                Me.Adapter.InsertCommand.Parameters(0).Value = CType(Product_ID.Value,I
            Else
                Me.Adapter.InsertCommand.Parameters(0).Value = Global.System.DBNull.Va
            End If
            If (Product_Name Is Nothing) Then
                Me.Adapter.InsertCommand.Parameters(1).Value = Global.System.DBNull.Va
            Else
                Me.Adapter.InsertCommand.Parameters(1).Value = CType(Product_Name,Stri
            End If
            If (Product_Description Is Nothing) Then
                Me.Adapter.InsertCommand.Parameters(2).Value = Global.System.DBNull.Va
            Else
                Me.Adapter.InsertCommand.Parameters(2).Value = CType(Product_Descriptio
            End If
```

```
            Dim previousConnectionState As Global.System.Data.ConnectionState = Me.Adap
            If ((Me.Adapter.InsertCommand.Connection.State And Global.System.Data.Conne
                        <> 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
                    Me.Adapter.InsertCommand.Connection.Close
                End If
            End Try
        End Function
    End Class
End Namespace
```

DataSet1.xsc

```
<?xml version="1.0" encoding="utf-8"?>
<!--<autogenerated>
     This code was generated by a tool.
     Changes to this file may cause incorrect behavior and will be lost if
     the code is regenerated.
</autogenerated>-->
<DataSetUISetting Version="1.00" xmlns="urn:schemas-microsoft-com:xml-msdatasource">
  <TableUISettings>
  </TableUISettings>
</DataSetUISetting>
```

DataSet1.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="DataSet1" targetNamespace="http://tempuri.org/DataSet1.xsd" xmlns:mstns=
  <xs:annotation>
    <xs:appinfo source="urn:schemas-microsoft-com:xml-msdatasource">
      <DataSource DefaultConnectionIndex="0" FunctionsComponentName="QueriesTableAdapte
        <Connections>
          <Connection AppSettingsObjectName="Settings" AppSettingsPropertyName="example
          </Connection>
        </Connections>
```

```
        <Tables>
          <TableAdapter BaseClass="System.ComponentModel.Component" DataAccessorModifie
            <MainSource>
              <DbSource ConnectionRef="example_dbConnectionString (Settings)" DbObjectN
                <InsertCommand>
                  <DbCommand CommandType="Text" ModifiedByUser="False">
                    <CommandText>INSERT INTO [dbo].[Products] ([Product ID], [Product N
                    <Parameters>
                      <Parameter AllowDbNull="True" AutogeneratedName="" DataSourceName
                      </Parameter>
                      <Parameter AllowDbNull="True" AutogeneratedName="" DataSourceName
                      </Parameter>
                      <Parameter AllowDbNull="True" AutogeneratedName="" DataSourceName
                      </Parameter>
                    </Parameters>
                  </DbCommand>
                </InsertCommand>
                <SelectCommand>
                  <DbCommand CommandType="Text" ModifiedByUser="False">
                    <CommandText>SELECT [Product ID], [Product Name], [Product Descript
                    <Parameters>
                    </Parameters>
                  </DbCommand>
                </SelectCommand>
              </DbSource>
            </MainSource>
            <Mappings>
              <Mapping SourceColumn="Product ID" DataSetColumn="Product ID" />
              <Mapping SourceColumn="Product Name" DataSetColumn="Product Name" />
              <Mapping SourceColumn="Product Description" DataSetColumn="Product Descr
            </Mappings>
            <Sources>
            </Sources>
          </TableAdapter>
        </Tables>
        <Sources>
        </Sources>
      </DataSource>
    </xs:appinfo>
  </xs:annotation>
  <xs:element name="DataSet1" msdata:IsDataSet="true" msdata:UseCurrentLocale="true" ms
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Products" msprop:Generator_UserTableName="Products" msprop:Ge
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Product_x0020_ID" msprop:Generator_UserColumnName="Proc
              <xs:element name="Product_x0020_Name" msprop:Generator_UserColumnName="Pr
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:maxLength value="100" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="Product_x0020_Description" msprop:Generator_UserColumnN
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:maxLength value="255" />
                  </xs:restriction>
                </xs:simpleType>
```

```
              </xs:element>
              <xs:element name="BarCode" msprop:Generator_UserColumnName="BarCode" mspr
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

DataSet1.xss

```
<?xml version="1.0" encoding="utf-8"?>
<!--<autogenerated>
     This code was generated by a tool to store the dataset designer's layout informat
     Changes to this file may cause incorrect behavior and will be lost if
     the code is regenerated.
</autogenerated>-->
<DiagramLayout xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http:/
  <Shapes>
    <Shape ID="DesignTable:Products" ZOrder="1" X="152" Y="169" Height="153" Width="19
  </Shapes>
  <Connectors />
</DiagramLayout>
```

Form1.Designer.vb

```
Partial Class Form1
        Private components As System.ComponentModel.IContainer = Nothing

        Protected Overloads Overrides Sub Dispose(disposing As Boolean)
                If disposing AndAlso (components IsNot Nothing) Then
                        components.Dispose()
                End If
                MyBase.Dispose(disposing)
        End Sub

        #Region "Windows Form Designer generated code"

        Private Sub InitializeComponent()
                Me.crystalReportViewer1 = New CrystalDecisions.Windows.Forms.CrystalRep
                Me.CrystalReport11 = New ReportFromSqlServer.CrystalReport1()
```

```
                Me.SuspendLayout()
                '
                ' crystalReportViewer1
                '
                Me.crystalReportViewer1.ActiveViewIndex = 0
                Me.crystalReportViewer1.BorderStyle = System.Windows.Forms.BorderStyle
                Me.crystalReportViewer1.Dock = System.Windows.Forms.DockStyle.Fill
                Me.crystalReportViewer1.Location = New System.Drawing.Point(0, 0)
                Me.crystalReportViewer1.Name = "crystalReportViewer1"
                Me.crystalReportViewer1.ReportSource = Me.CrystalReport11
                Me.crystalReportViewer1.Size = New System.Drawing.Size(799, 566)
                Me.crystalReportViewer1.TabIndex = 0
                '
                ' Form1
                '
                Me.AutoScaleDimensions = New System.Drawing.SizeF(6F, 13F)
                Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
                Me.ClientSize = New System.Drawing.Size(799, 566)
                Me.Controls.Add(Me.crystalReportViewer1)
                Me.Name = "Form1"
                Me.Text = "Form1"
                Me.ResumeLayout(False)

        End Sub

        #End Region

        Private crystalReportViewer1 As CrystalDecisions.Windows.Forms.CrystalReportVie
        Private CrystalReport11 As CrystalReport1


End Class
```

Form1.vb

```
Imports System.Data
Imports System.Diagnostics
Imports System.Windows.Forms
Imports System.Data.SqlClient
Imports Bytescout.BarCode

Public Partial Class Form1
        Inherits Form
        Public Sub New()
                InitializeComponent()

                Try
                        ' MODIFY THE CONNECTION STRING WITH YOUR SERVER CONNECTION INF
                        Const  connectionString As String = "Data Source=localhost\SQL
```

```
                Using connection As New SqlConnection(connectionString)
                        connection.Open()

                        ' Create a database for demonstration purposes
                        '''//////////////////////////////////////////////////

                        Dim o As [Object] = ExecuteQueryScalar(connection, "SE

                        ' if 'example_db' does not exist, create it
                        If o Is Nothing OrElse TypeOf o Is DBNull Then
                                ' Create empty database
                                ExecuteQueryWithoutResult(connection, "CREATE D
                                ' Switch to created database
                                ExecuteQueryWithoutResult(connection, "USE exa
                                ' Create a table
                                ExecuteQueryWithoutResult(connection, "CREATE T
                                ' Fill the table with data
                                ExecuteQueryWithoutResult(connection, "INSERT I
                                ExecuteQueryWithoutResult(connection, "INSERT I
                                ExecuteQueryWithoutResult(connection, "INSERT I
                                ExecuteQueryWithoutResult(connection, "INSERT I
                        End If


                        ' Create a dataset from query.
                        ' Query result columns must conform to field names we u

                        Dim dataAdapter As New SqlDataAdapter("SELECT [Product

                        ' fill dataset
                        Dim dataSet As New DataSet()
                        dataAdapter.Fill(dataSet)

                        ' don't forget to close the connection
                        connection.Close()

                        ' add virtual column into the result table
                        dataSet.Tables(0).Columns.Add(New DataColumn("BarCode"

                        ' create barcode object
                        Dim bc As New Barcode(SymbologyType.Code39)
                        bc.DrawCaption = False

                        For Each row As DataRow In dataSet.Tables(0).Rows
                                ' set barcode value
                                bc.Value = (Convert.ToString(row("Product ID"))

                                ' retrieve generated image bytes
                                Dim barcodeBytes As Byte() = bc.GetImageBytesWM

                                ' fill virtual column with generated image byte
                                row("BarCode") = barcodeBytes
                        Next

                        ' set report datasource
                        CrystalReport11.SetDataSource(dataSet.Tables(0))
                End Using
        Catch ex As Exception
                Trace.WriteLine("Error: " & ex.Message)
```

```
                End Try
        End Sub

        Private Shared Sub ExecuteQueryWithoutResult(connection As SqlConnection, query
                Using command As New SqlCommand(query, connection)
                        command.ExecuteNonQuery()
                End Using
        End Sub

        Private Shared Function ExecuteQueryScalar(connection As SqlConnection, query A
                Using command As New SqlCommand(query, connection)
                        Return command.ExecuteScalar()
                End Using
        End Function
End Class
```

Program.vb

```
Imports System.Collections.Generic
Imports System.Windows.Forms

NotInheritable Class Program
        Private Sub New()
        End Sub
        <STAThread> _
        Friend Shared Sub Main()
                Application.EnableVisualStyles()
                Application.Run(New Form1())
        End Sub
End Class
```

VIDEO

https://www.youtube.com/watch?v=REnj3A-oSPI

ON-PREMISE OFFLINE SDK

60 Day Free Trial or Visit ByteScout Barcode SDK Home Page
Explore ByteScout Barcode SDK Documentation
Explore Samples
Sign Up for ByteScout Barcode SDK Online Training

ON-DEMAND REST WEB API

Get Your API Key
Explore Web API Docs
Explore Web API Samples

visit www.ByteScout.com

visit www.PDF.co