Inventory Integration for Southern Marsh Co.

December 9, 2012

Ravikumar Chimmalgi

Table of Contents

1. **Introduction** ---------- 3

2. **Project Goals** ---------- 3

3. **Shopify** ---------- 3

4. **Fishbowl** ---------- 4

5. **API Methods for Web service** ---------- 4

    5.1. Shopify ---------- 4

        5.1.1 Get Orders ---------- 4

        5.1.2 Update Product Quantity ---------- 4

        5.1.3 Get Order Details ---------- 6

    5.2. Fishbowl ---------- 7

        5.2.1 Add Sales Order ---------- 7

        5.2.2 Get Sales Order ---------- 8

6. **Process Flow** ---------- 9

1. Introduction

Southern Marsh is a local apparel company which uses Shopify for their online store and Fishbowl for their onsite inventory management and invoicing. The problem with using the two systems is that there is no integration between them. Currently, there is a two-step process to keep both the systems concurrent.

For adding new items to Shopify, they import the updated inventory from Fishbowl in the form of a CSV file and repopulate the Shopify database.

To update their Fishbowl inventory, they print orders from Shopify and manually input them in Fishbowl order list. This automatically decreases the number of items in their Fishbowl inventory according to the orders placed on Shopify. They do this once per day. As a consequence, often there are cases where more is sold during the day than they have in stock.

For sales analysis, Southern Marsh gets the sales data and uses excel to keep track of performance. But the problem with this is that the process is lengthy, provides inaccurate sales prediction and there is no way of knowing how fast a certain item is being sold so they can keep more of that item in stock.

2. Project Goals

1) To create a web service application which integrates Fishbowl and Shopify by using the respective APIs. The service application will routinely get the orders from Shopify and automatically update Fishbowl inventory & orders and also periodically update Shopify quantities due to Fishbowl orders.

2) To provide better sales analytics capability and more accurate forecasting by creating a data warehouse and using custom cubes for analysis.

3. Shopify

Shopify is a platform which allows its customers to create online stores. Shopify provides the ability to set up simple apps (using Ruby on Rails), as well as an API and other tools for pulling down and pushing up data to the site.

Shopify provides an API which supports RESTful communication.

4. Fishbowl

Fishbowl is inventory management software which integrates with QuickBooks for most of the financial functions (like invoicing). Primary functionality Fishbowl provides is inventory control, purchase order & sales order generation, manufacturing, and cycle counting.

Fishbowl also supports RESTful communication.

5. API Methods for Web Service

5.1 Shopify

5.1.1 GetSalesOrders

This method gets all the orders created after a specific order. The orders ids increase in number so its easy to keep track of the ids and only get new orders.

```
Public Function getOrders() As String

        Dim client As New WebClient
        client.Credentials = New NetworkCredential(api_key, password)

        Dim stream As IO.Stream =
client.OpenRead("https://smwebserviceproject.myshopify.com/admin/orders.xml?since_id=1505
83138")
        Dim reader As New StreamReader(stream)
        Dim s As String = reader.ReadToEnd


        Return s
    End Function
```

5.1.2 Update Product Quantity

This method is for updating Shopufy inventory due to orders from retailers on Fishbowl. It takes a list of all the variants to be updated with their Shopify variant ids and updates the variants in Shopify.

```
Public Sub updateShopifyVariants(ByVal variantlist As List(Of variantsToUpdate))

        For Each var In variantlist
            ' Create a request for the URL.
            Dim request As WebRequest =
WebRequest.Create("https://smwebserviceproject.myshopify.com/admin/variants/259224462.xml
")
            request.Credentials = New NetworkCredential(api_key, password)
            Dim buffer As System.Text.StringBuilder = New System.Text.StringBuilder("")
            buffer.Append("<?xml version=""1.0"" encoding=""UTF-8""?>")
            buffer.Append("<variant>")
            buffer.Append("<id type=""integer"">" + var.variantid + "</id>")
            buffer.Append("<inventory-quantity type=""integer"">" + var.quantity +
"</inventory-quantity>")
            buffer.Append("</variant>")

            Dim str As String = buffer.ToString()

            request.Method = "PUT"

            'important
            request.ContentType = "application/xml"


            ' Turn the XML into a byte buffer to prepare it for transmission
            Dim bytebuffer As Byte() = System.Text.Encoding.UTF8.GetBytes(str)
            request.ContentLength = bytebuffer.Length
            Dim streamData As Stream = request.GetRequestStream()
            streamData.Write(bytebuffer, 0, bytebuffer.Length)
            streamData.Close()


            Dim response As HttpWebResponse = Nothing
            Try
                ' This is where the HTTP POST actually occurs.
                response = DirectCast(request.GetResponse(), HttpWebResponse)
            Catch a As Exception
                Console.WriteLine(a.ToString())
            End Try
            ' Display the status.
            Console.WriteLine(response.StatusDescription)
            ' Get the stream containing content returned by the server.
            Dim dataStream As Stream = response.GetResponseStream()
            ' Open the stream using a StreamReader for easy access.
            Dim reader As New StreamReader(dataStream)
            ' Read the content.
            Dim responseFromServer As String = reader.ReadToEnd()
            ' Display the content.
            Console.WriteLine(responseFromServer)
            ' Cleanup the streams and the response.
            reader.Close()
            dataStream.Close()
            response.Close()

        Next
    End Sub
```

5.1.3 Get Order Details

This method takes the xml of the get order method and stores all the order data into a list for future use in making xml requests to add sales order in fishbowl.

```
Public Function getOrderDetails(ByVal xdoc As Xml.XmlDocument) As List(Of
shopifyorder)
        Dim orderlist As New List(Of shopifyorder)

       Dim orderNodeList As XmlNodeList = xdoc.SelectNodes("/orders/order")


        Console.WriteLine(orderNodeList.Count)
        For Each ordernode In orderNodeList

            Dim order As New shopifyorder
            order.orderid = ordernode.Childnodes.Item(7).innertext
            order.orderName = ordernode.Childnodes.Item(29).innertext
            order.createdAt = ordernode.Childnodes.Item(20).innertext
            order.totalPrice = ordernode.Childnodes.Item(16).innertext

            Dim billNodeList As XmlNodeList = xdoc.SelectNodes("/orders/order/billing-
address")
            For Each billNode In billNodeList
                order.billName = billNode.Childnodes.Item(12).innertext
                order.billAddress = billNode.Childnodes.Item(2).innertext + " " +
billNode.Childnodes.Item(3).innertext + " " + billNode.Childnodes.Item(4).innertext
                order.billCity = billNode.Childnodes.Item(4).innertext
                order.billZip = billNode.Childnodes.Item(9).innertext
            Next

            Dim shipNodeList As XmlNodeList =
xdoc.SelectNodes("/orders/order/shipping-address")
            For Each shipNode In shipNodeList
                order.shipName = shipNode.Childnodes.Item(12).innertext
                order.shipAddress = shipNode.Childnodes.Item(2).innertext + " " +
shipNode.Childnodes.Item(3).innertext
                order.shipZip = shipNode.Childnodes.Item(9).innertext
                order.shipCountry = shipNode.Childnodes.Item(6).innertext
                order.shipState = shipNode.Childnodes.Item(8).innertext
            Next

            Dim itemNodeList As XmlNodeList = xdoc.SelectNodes("/orders/order/line-
items/line-item")
            For Each itemNode In itemNodeList
                order.title = itemNode.Childnodes.Item(7).innertext
                order.fishbowlSku = itemNode.Childnodes.Item(6).innertext
                order.quantity = itemNode.Childnodes.Item(5).innertext

            Next
            Dim taxNode As XmlNode = xdoc.SelectSingleNode("/orders/order/tax-
lines/tax-line")
            order.taxname = taxNode.ChildNodes.Item(0).InnerText
            order.taxrate = taxNode.ChildNodes.Item(2).InnerText
            orderlist.Add(order)

        Next
        Return orderlist

    End Function
```

5.2 Fishbowl

5.2.1 Add Sales Order

This method adds sales orders to Fishbowl by creating the XML request using the data obtained from Shopify orders.

```
'AddSalesOrder.
    'Creates new sales order
    Public Sub AddSalesOrder(ByVal order As List(Of shopifyorder))
        '***implement
        For Each ord In order
            Dim ord As shopifyorder = order(0)
            Dim buffer As System.Text.StringBuilder = New
System.Text.StringBuilder("")
            buffer.Append("<FbiXml>" + ticket)
            buffer.Append("<FbiMsgsRq> " & _
                          "<AddSOItemRq>" & _
                          "<SalesOrder> " & _
                          "<Salesman></Salesman>")
            buffer.Append("<Number></Number>" & _
                          "<Status>10</Status>" & _
                          "<Carrier>FEDEX</Carrier>" & _
                          "<FirstShipDate></FirstShipDate>")

            buffer.Append("<CreatedDate>" + ord.createdAt + "</CreatedDate>")
            buffer.Append("<IssuedDate></IssuedDate>" & _
                          "<TaxRatePercentage>" + ord.taxrate + "</TaxRatePercentage>"
& _

                          "<TaxRateName>" + ord.taxname + "</TaxRateName>" & _
                          "<ShippingTerms></ShippingTerms>" & _
                          "<PaymentTerms>COD</PaymentTerms>" & _
                          "<CustomerContact></CustomerContact>" & _
                          "<CustomerName>SouthernMarshCustomer</CustomerName>" & _
                          "<FOB></FOB>" & _
                          "<QuickBooksClassName></QuickBooksClassName>" & _
                          "<LocationGroup></LocationGroup>")

            buffer.Append("<BillTo>" & _
                          "<Name>" + ord.billName + "</Name>" & _
                          "<AddressField>" + ord.billAddress + "</AddressField>" & _
                          "<City>" + ord.billCity + "</City>" & _
                          "<Zip>" + ord.billZip + "</Zip>" & _
                          "</BillTo>")

            buffer.Append("<Ship>" & _
                          "<Name>" + ord.shipName + "</Name>" & _
```

```
                            "<AddressField>" + ord.shipAddress + "</AddressField>" & _
                            "<Zip>" + ord.shipZip + "</Zip>" & _
                            "<Country>" + ord.shipCountry + "</Country>" & _
                            "<State>" + ord.shipState + "</State>" & _
                            "</Ship>" & _
                            "</SalesOrder>")

              'loop for each salesorderitem
              buffer.Append("<SalesOrderItem>" & _
                            "<ID>1</ID>" & _
                            "<ProductNumber>" + ord.fishbowlSku + "</ProductNumber>" & _
                            "<SOID>2</SOID>" & _
                            "<Description>" + ord.title + "</Description>" & _
                            "<Taxable>true</Taxable>" & _
                            "<Quantity>" + ord.quantity + "</Quantity>" & _
                            "<ProductPrice>" + ord.totalPrice + "</ProductPrice>" & _
                            "<TotalPrice>" + ord.totalPrice + "</TotalPrice>" & _
                            "<UOMCode>ea</UOMCode>" & _
                            "<ItemType>10</ItemType>" & _
                            "<Status>10</Status>" & _
                            "<QuickBooksClassName></QuickBooksClassName>" & _
                            "<NewItemFlag>false</NewItemFlag>" & _
                            "</SalesOrderItem>")

              buffer.Append("</AddSOItemRq >" & _
                            "</FbiMsgsRq>" & _
                            "</FbiXml>")

              Console.WriteLine("Buffer: " + buffer.ToString)

              Dim DS As New DataSet
              DS.ReadXml(XmlReader.Create(New
StringReader(IssueCommand(buffer.ToString))))


          Next


      End Sub
```

5.2.2 Get Sales order

This function returns all the sales order created after the DateCreatedBegin variables.

```
Public Function GetSalesOrder() As DataSet
 Dim S As String = "<FbiXml>" & ticket & _
                   "<FbiMsgsRq><GetSOListRq>" & _
                  "<DateCreatedBegin>2012-12-09T00:00:00</DateCreatedBegin>" & _
                   "</GetSOListRq></FbiMsgsRq></FbiXml>"
 Dim DS As New DataSet
 DS.ReadXml(XmlReader.Create(New StringReader(IssueCommand(S))))
 Return DS
```

```
    End Function
```

6. Process Flow

1) For the first time running the service, a current Shopify order number should be given and also starting time for fishbowl order is needed.

2) After every hour, the service uses gets orders from shopify and stores all the order details in a list

3) Then Fishbowl is authenticated and new sales order is added using the saved shopify order details in the list. These orders would then be handled the same way Fishbowl orders are handled and the inventory will be updated when they are fulfilled and shipped. All the Shopify orders are created under the name SouthernMarshCustomer

4) Then all new orders after the starting time specified are obtained from fishbowl. Then all the orders that are not of SouthernMarshCustomer are stored in a list. This list is updated with shopify ids by cross referencing a database consisting of all the Fishbowl SKU and corresponding Shopify variant ids.

5) The variant quantities of Shopify is updated by using the list of all the variants-to-be-updated by using the Shopify variant id obtained from the database and quantity obtained from the orders.