Software Code Performance Review

[redacted]

Table of Contents

1 Executive Summary

[redacted] was developed using Active Server Pages, commonly known as Classic ASP, which was released over 20 years ago by Microsoft. As with any new technology at the time, there was not a lot of emphasis on performance and scalability which can be found in lots of successor technologies today.

Over time applications developed with Classic ASP begin to experience latency and other performance issues due to human, process and technology factors. These issues are evident in a mature web site application like [redacted].

Your developers have gone to great lengths to optimize the code, but the performance is still not where it should be in today's ecommerce world. The purpose of this project is to identify areas with the Classic ASP code of [redacted] where page speed can be increased. Our focus was to identify a list of short term fixes. A few longer term suggestions are included at the end.

2 Scope of Review

[redacted] contains a lot of code and functionality that has been written and re-written many times. Much of the site code consists of HTML files which are generated by the IT group as needed. By generating HTML, they remove calls to the database. The focus of this phase was to look at code related to the e-commerce process. Based on what we heard, and our own tests, these pages are particularly slow since they access the database much more so than the other pages. The following were covered:

- Home page
- Product list
- Product category pages
- Product detail pages
- Product search page
- Shopping cart pages

3 Recommendations for Short Term Fixes

3.1 Page/Scripting Issues

A cursory review of select ASP pages reveals that some recommended best practices in developing ASP applications have not been widely followed.

3.1.1 Missing Response Buffering

Response buffering is not enabled in most ASP pages reviewed. This results of not buffering is that every Response.Write is being sent to the client immediately, which is inefficient especially when processing large amount of data.

3.1.2 Not Using "Option Explicit"

ASP execution engine will create a new variable when a variable name is misspelled leading to undesirable results on page/business logic. Setting this declaration at the top of ASP pages will force the user to declare all the used variables and it will improve performance slightly.

3.1.3 Unnecessary context switching

In some files, the tags `%><%` were used to create separation between function and sub routines. These tags between functions and sub routines are unnecessary as the create context switching between the script engine and HTML during the execution of the script thereby reducing performance.

Example page:

- incpages/AjaxReloadShippingOption.asp
- incpages/AjaxReloadSummaryPlaceOrder.asp

3.2 Database Queries

There are several instances where database operations are not optimally implemented. The following pages, containing the include file **\incpages\SQLInject.asp,** were reviewed

- \incpages\AjaxCaBi-AB.asp
- \incpages\ajaxfilterbrand_new.asp
- \incpages\AjaxGetHTML.asp
- \incpages\AjaxMatchCC.asp
- \incpages\AjaxPlaceOrder-AddressValidation.asp
- \incpages\AjaxPlaceOrder.asp
- \incpages\AjaxShippingMethodChosen_dropship_v3.asp
- \incpages\AjaxUPW.asp
- \incpages\AjxDropShipOrder.asp
- \incpages\CheckHomepageAlert_canada.asp
- \incpages\CheckStockStatusNew.asp
- \incpages\CountryPopup.asp
- \incpages\findorder.asp
- \incpages\OrderFinished_SP26665.asp
- \incpages\Send-Email-Welcome-Off-10.asp
- \incpages\URLrewrite_404tohome.asp
- \incpages\ViewAllColors.asp
- \incpages\ViewAllColors_SP24491.asp
- \incpages\ViewAllColorsNew.asp
- \incpages\ViewAllColorsPopup.asp

3.2.1 Query scripts using wildcard (*)

Using wildcards in SQL queries have performance implications as the result set returned contains all data columns along with unnecessary data. There were many examples of the “select *” in the ASP files Example file:

- incpages\PlaceOrder.asp

```
SQL = "SELECT * FROM Shipping with(nolock) WHERE Shipping.ShipCHOSEN = 'Y' AND Ship
Call OpenVRS(RS_Shipping1,"RS_Shipping1",SQL) : If NOT IsObject(RS_Shipping1) Then

'SQL = "SELECT * FROM Billing  WHERE Billing.BillingCHOSEN = 'Y' AND Billing.Custom
'Call OpenVRS(RS_Billing1,"RS_Billing1",SQL) : If NOT IsObject(RS_Billing1) Then Er

SQL = "SELECT * FROM CCards with(nolock) ,PaymentMethodsAmazon PaymentMethods with(
Call OpenVRS(RS_CCards1,"RS_CCards1",SQL) : If NOT IsObject(RS_CCards1) Then Err.Ra


If NOT RS_Shipping1.EOF Then
    Session("APOAddress") = ""
    if trim(Lcase(RS_Shipping1.fields("ShipCity"))) = "apo" or trim(Lcase(RS_Shippi
        Session("APOAddress") = "Y"
    end if

    pShipState = RS_Shipping1.fields("ShipState") ''pShipState : ShippingFunctions_
    pShipPostalCode = RS_Shipping1.fields("ShipPostalCode")
    If Request.QueryString("cuong") = "Y" Then
        Response.Write("<br>pShipPostalCode =" &  pShipPostalCode)
    End If
    Session("ShipState") = pShipState ''added 04-25-2014

    If lcase(RS_Shipping1.Fields("ShipCountry")) = "canada" and Request.Cookies("US
```

In the above example, the Shipping or CCards Table may contain and return data columns that are not required when executing the **incpages\PlaceOrder.asp** page. The size of data returned in a record set varies based on the number of columns requested and the amount of data in those columns hence using wildcard characters in SQL queries impacts on performance. In a low traffic site this might work, but can eat up memory in a site like [redacted].com with traffic peaks of 4,000+ concurrent connections.

3.2.2 Column referenced by name in record set

While this is common practice because of code readability, referencing columns by ordinal (the column position index starting from 0) provides up to 50% increase in performance.

Recordsets opens a connection to the database, it then calls a query which opens the recordset and leaves it open. Then you move through the recordset (by the movenext command) and when you want you can call specific fields to display. When you have finished you close the recordset and close the connection.

Getrows first starts by opening a connection to the database and calling a query which loads a recordset. Then the getrows command pulls the entire contents of the recordset into a two-dimensional array. We then close the recordset and the database connection as we no longer need it. We are now working from the memory using the array variable.

Our recommendation is to select the most frequently accessed pages and consider referencing columns by ordinal instead of by name in a record set. What is lost in code readability is gained in performance; with affected pages loading up to 50% faster. However, in order to make the code readable again, variables can be declared and assigned indexes instead of just using numerals to access the array data.

Example code:

```
Const user_name = 0
Const user_email = 1
Const user_tel = 2

DIM users_eof = true

SET Conn = Server.CreateObject("ADODB.Connection")
Conn.Open DBName
SET RS = Conn.execute(STORED_PROCEDURE)

IF NOT RS.EOF THEN
    users = RS.getrows()
    users_eof = false
END IF

RS.Close
Conn.Close
SET Conn = nothing

IF NOT users_eof THEN
    FOR record = 0 to UBound(users, 2)
        response.write("Name: " & users(user_name, record) & "")
        response.write("Email: " & users(user_email, record) & "")
        response.write("Telephone: " users(user_tel, record) & "
    NEXT
END IF
```

3.2.3 Unnecessary include files

Some files are included in pages even if they aren't needed. For example, there are pages that don't contain any database operations that include the SQLInject.asp file. This unnecessary include gets executed every time the page loads, adding additional overhead to page processing. Example files

- incpages\AjaxShippingMethodChosen_dropship_v3.asp
- incpages\AjaxShippingMethodChosen.asp

```
<!-- #include file="../incfiles/SQLInject.asp" -->
<%
    Dim strSession
    if Request.Form("RemoveDSC") = "Y" then
        Session("CouponCode") = ""
        Session("CouponCode") = ""
        Session("OLD_CouponCode") = ""
        Response.Cookies("NONE_EXPIRES").Expires = Date() - 365
        Response.Write("CouponCode[" & Session("CouponCode") & "]")
    else
        strSession = Request.QueryString("session") & ""
        if strSession = "" Then
            strSession = "ShippingSpeedChoice"
        End If

        if Request.QueryString("id") <> "" then
            Session(strSession) = Request.QueryString("id")
            Response.Cookies(strSession) = Session(strSession)
            Response.Cookies(strSession).Expires = Date() + 1

            if Session(strSession) = "" OR Session(strSession) = "0" then
                Session(strSession) = Request.Cookies(strSession)
                Response.Cookies(strSession) = Session(strSession)
                ''Response.Cookies(strSession).Domain = ".swimoutlet.com"
                Response.Cookies(strSession).Expires = Date() + 1
            end if
        Else
            Session(strSession) = ""
            Response.Cookies(strSession) = ""
            Response.Cookies(strSession).Expires = Date() - 999
        end if

        Response.Write(Session(strSession))
    end if
%>
```

In the example above, the file **incpages\AjaxShippingMethodChosen.asp** does not access the database however it includes the **incfiles\SQLInject.asp** which adds an overhead to the page processing.

3.2.4 Database connection not closed

Some of the files contain database connection that may not be closed because the close method of the database connection variable (e.g. conn.close()) is contained inside conditional statements that may not always be executed. Example files

- incpages\AjaxGetHTML.asp

In the example image below, the database connection is set to open at the beginning of the page execution.

```
<!-- #include file="../incfiles/SQLInject.asp" -->
<!-- #include file="../../ConnString.asp" -->
<!-- #include file="../incfiles/functions.asp" -->
<!-- #include file="../incfiles/adovbs.inc" -->
<!-- #include file="../incfiles/functions_ShippingAddress_RC.asp"-->
<%
Dim Conn
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.ConnectionTimeout = 20
Conn.Open Config_ConnectionString
%>
<!--#include file="../incfiles/configsetup.asp" -->
<%
Function NUMBER_SQLINJECTION(byval str)
    dim str_old
    str_old = Trim(str)
    str = Trim(LCase(str))
    str = Replace(str," ","")
    str = Replace(str,"select","")
    str = Replace(str,"from","")
    str = Replace(str,"tables","")
    str = Replace(str,"table","")
    str = Replace(str,"or","")
    if LCase(str_old) = LCase(str) then
        str = str_old
    end if
    NUMBER_SQLINJECTION = str
End Function
Function STRING_SQLINJECTION(byval str)
    dim str_old
    str_old = Trim(str)
    str = Trim(LCase(str))
    ''str = Replace(str," ","")
    str = Replace(str,"select","")
    str = Replace(str,"from","")
    str = Replace(str,"tables","")
    str = Replace(str,"table","")
    str = Replace(str,"or","")
    if LCase(str_old) = LCase(str) then
        str = str_old
    end if
    STRING_SQLINJECTION = Trim(str)
End Function
'''-----------------------------------------------------------------------------------------------------------
'''-----------------------------------------------------------------------------------------------------------
'''-----------------------------------------------------------------------------------------------------------
    DIM action
    action = NUMBER_SQLINJECTION(Request.QueryString("action"))
```

The page executes various functions and statements on the page if the condtion for execution is met. If the conditions are not met, the page is processed to the end without executing those functions. If the connection to the datbase is still open, it will remain open because it has not been explicitly closed at the bottom of the page. See image below:

```
            count_div_clear = count_div_clear + 1
            if (count_div_clear mod 3) = 0 then
            %>
                <div class="clear"></div>
            <%
            end if
            Wend
            %>
        </div>
        <%
        end if

        RS_Billing1.Close()
        set RS_Billing1 = nothing
        %>
        <!--</div>-->
        <%
    end if ''END PLIST
'''-----------------------------------------------------------------------------------------------------------
'''-----------------------------------------------------------------------------------------------------------
'''-----------------------------------------------------------------------------------------------------------
%>
```

3.2.5 Database connection opened outside of conditional statement

There are instances where the database connection objects are created and opened outside of conditional statements. In these instances, if those conditional statements are not met then the database connections will remain open. Example pages:

- incpages\AjaxUPW.asp
- incpages\AjaxMatchCC.asp

```
<%
Session.CodePage=65001
Response.Charset="UTF-8"
%>
<!-- #include file="../incfiles/SQLInject.asp" -->
<!-- #include file="../../ConnString.asp" -->
<!-- #include file="../incfiles/functions.asp" -->
<!-- #include file="../incfiles/adovbs.inc" -->
<!-- #include file="../incfiles/functions_Checkout.asp"-->
<%
If Request.ServerVariables("CERT_KEYSIZE") <> "" Then
    'GOOD, we are secure
Else
    Response.End()
End If
%>
<%
Dim Conn
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.ConnectionTimeout = 20
Conn.Open Config_ConnectionString


Private Function HTMLDecode(byVal encodedstring)
    Dim tmp, i
    tmp = encodedstring
    tmp = Replace( tmp, "&quot;", chr(34) )
    tmp = Replace( tmp, "&lt;"  , chr(60) )
    tmp = Replace( tmp, "&gt;"  , chr(62) )
    tmp = Replace( tmp, "&amp;" , chr(38) )
    tmp = Replace( tmp, " ", chr(32) )
    For i = 1 to 255
        tmp = Replace( tmp, "&#" & i & ";", chr( i ) )
    Next
    HTMLDecode = tmp
End Function
%>
<!--#include file="../incfiles/configsetup.asp" -->
<%
    dim RS_Query1,strQuery,arrName,sFirstName,sLastName,SQL_E,custId,cardid,
    dim Edit_CreditCardType,CreditCardNumber,cid,CardHoldersName,ExpDate
    set cmd = server.createobject("ADODB.Command")
    '--------------------------------------------------------------------
    if Request.QueryString("action") = "GC" AND Request.QueryString("argc")
        set RS_Query1 = Server.CreateObject("ADODB.Recordset")
        strQuery = ""
```

4

Longer Term Recommendations

4.1 Remove SQL scripts embedded in page code

There is a lot of SQL code contained in the ASP pages. Moving these to the database and optimizing would improve performance. It would also help lower the maintenance resources required; storing database code in ASP files is not as easily reusable because the business logic is not centralized. The current implementation also creates a vulnerability which could make it easier for a hacker to carry out an SQL injection attack. Beyond performance some of the benefits of using stored procedures include but are not limited to:

- Stored procedures are compiled on first run by the SQL server with the server creating a query plan. Subsequent calls are much faster due to the cached query plan being executed by the server.
- Encapsulated code can be reused.
- Easy to maintain and update without code deployment
- Not prone to SQL injection attacks.

4.2 Improve database capabilities

Based on the code review and conversations with [redacted], there is a mindset to put as little strain on the database. Improving performance in that area would allow more logic to be moved from the ASP pages.

4.3 Upgrade to .NET or other modern technology

Modern web pages are heavy and require a lot of processing which are just not feasible with Classic ASP. Beyond performance gains, scaling, multi-language and multithreaded applications, security, reusable code, portability, maintainability, SEO and a host of other features would be much easier to achieve.

Changing the entire site would be a large project. Our recommendation would be to explore options that would leave the majority of the product information and list pages as is, but to set up a new URL such as buy.[redacted].com for your e-commerce process.

5 References

5.1 Implicit and Explicit Declaration (of variables)

https://msdn.microsoft.com/en-us/library/xe53dz5w(v=vs.100).aspx

5.2 Recordset vs. Getrows

https://www.globalgraphics.co.uk/blog/classic-asp--recordsets-vs-getrows.html

5.3 Database connection and effects on performance

https://www.brentozar.com/archive/2014/05/connections-slow-sql-server-threadpool/

5.4 25+ ASP Tips to Improve Performance and Style

https://msdn.microsoft.com/en-us/library/ms972335.aspx