

Architecture

Overview

EquipmentBarn.com (known as EB from here on) is built almost exclusively using the Microsoft .Net framework. The webserver is Microsoft IIS 6.0 running on a Windows Server 2000 server. The database is currently Microsoft SqlServer2000. The only Microsoft platform exception is the email notifications which are executed using a QMail v2.1 mail server running the RedHat Linux 7.2 OS.

Communication between the Linux email system and the Microsoft application system is facilitated through the use of web services. The web service is used by perl scripts on the Linux box to retrieve and update the necessary email notifications from the email notifications table in the main database. Web Service requests are made from the Windows 2000/IIS Server to the Linux server for real time email needs-currently only used by forgotten password requests.

Graphics used in the site and marketing materials were completed using Adobe Photoshop 6.0 and Adobe Illustrator 6.0. Brochures and flyers were completed using Adobe PageMaker 7.0. Various diagrams and database schematics were completed using Microsoft Visio2000. Source Code management is handled using Microsoft Visual Source Safe 6.0.

EB was built in large part as an excuse to build a full featured website using Microsoft's .Net technologies. Many of the approaches have been iterative as I learned new methods within the framework. I tried to include as many of the major areas of .Net as could be reasonably applied to the project and to learn and demonstrate as much of the set as possible.

UI Elements and building

The headers on all of the pages are based off of a custom developed User Control. It uses values stored in the session and the current page name to determine what elements should be drawn. While this makes for a nice UI experience it really inhibits the caching ability...especially for brochureware. One example is if the user is logged in we want to show a logout element and vice versa. We can not cache any pages because we never really know what the state of the next user will be. I am going to look into section caching at a later time because I think it really could speed the whole site up a lot.

The footer is a standard HTML include that is included on all pages. It may be best to move this into the IIS standard footer section at a later time.

Database Access/Stored Procedures

The database access takes place through a data access layer that primarily relies on SQL Server stored procedures written in TSQL. When the next version of SQL Server

(Yukon) is released I anticipate rewriting most of the stored procs in C# or maybe VB.Net.

While storing a lot of the logic in the database ties the architecture pretty closely to SQL Server there is a Data Access Layer that resides in the application and does some conversion to Reader Objects or Data Sets. This will make it relatively easy to port to Oracle or some other database engine if necessary. If I end up using MySQL or something that does not support stored procedures there will have to be quite a bit of code added to the Data Access Layer.

How to run a high performing site over a midrange DSL line

One of the main concerns I faced bringing up this site is performance over a midrange DSL line. Being a site that supports the ability for users to upload their own images and download images of products the potential for moving large amounts of data exists.

In order to minimize this I have implemented a method of storing the root directory of all images as a URL. This allows me to set the root dir at application start so that the application can dynamically change the location. This will allow me to run and develop the site on a local machine and have it be fast or to take it on a laptop to demonstrate the site. However for the production system I have leased space of a set of 3rd party servers that have T1 connection speeds. All uploaded images on the production system will be stored there and referenced via the defined URL in the application. There should be no indication to the users that the images are coming from another server and the volume of traffic required on the local DSL line should be greatly reduced!

The only other adjustment made for speed was to turn on HTTP 1.1 compression. If the users browser is incapable of supporting HTTP 1.1 compression it will fall back to standard HTTP for that session.

Email Integration

As mentioned earlier the email for Equipment Barn is handled by Qmail server running on a Red Hat Linux box. The integration could probably have been a lot simpler had I used Microsoft Exchange and ran the whole system on a single server. However running on 2 separate boxes and platforms gave me an excuse to develop a web service and facilitates an abstracted approach to the email so that if I ever do decide to replace the email system it should be very simple.

Other benefits of using the Qmail system are its security, stability, and speed. If I ever decide to co-locate the webserver somewhere else using a web service will allow me to keep the email in house without requiring me to pay the fees associated with co-locating two machines. Another benefit is the removing the dependency on the two systems. If the email system were to go down EquipmentBarn would still be able to function and queue its messages until the email system is back up.

The way this works is that all emails sent from EB are queued into an Email table. The initial status of these messages is "Queued". There is a web service listener running on the Equipment Barn server, this is a completely separate program and process from the Equipment Barn website. The Linux/Qmail box has a Cron job (a Unix scheduled task) written in Perl that will run on scheduled intervals and make a web service request for all "Queued" mail. The Equipment Barn web service listener will send the mail to the Linux box and change the status to "Sent" and the Linux job will then take the email and inject it into the Qmail queue.

The only missing piece in this chain is for "bounced" or undeliverable messages. There is another cron job that runs on the Linux box and watches the Qmail message logs. If it sees a bounced message it will make another web service call to the EB server and the EB listener will change the message status to "Undeliverable" and may be retried at a later date as appropriate.

Most of the messages sent out from EB are informational and no responses are expected. However there are mailboxes set up for all sender accounts which are all directed to the main administrator box should someone respond.

Many of the messages (such as notices about an ad expiring) contain links, which a user can click on to renew or cancel the listing in question. These will all be directed to the appropriate detail page for viewing or in the case of any updates all will go to the EmailResponder.aspx page, there will always be at least two URL parameters on these requests. First the GUID (Globally Unique Identifier), which can be used to match the response up to the message. The 2nd will be the message type being responded to - drives the processing in the Email Responder system. In addition there may be various other parameters such as if the item should be renewed or deleted as in the example above.

Email Templates and Replaceable Parameters

One of the larger challenges and more complex pieces of EB has been making friendly, customized, emails. The way this was accomplished was through the use of templates and replaceable parameters. All the email templates are stored in the database. The templates contain various tags such as <USER-NAME>, <AUCTION-DATE>, <MESSAGE-GUID>, or <EQUIPMENT-CITY>. When a request is made for unsent email (from the Linux web service) the template will be parsed and these strings will be replaced with the appropriate values for the current user, equipment piece in question, auction, etc.

The basic tables involved are :

EB_EMAIL – Hold the to, from, values, etc. The subject and text fields will contain only the tag delimited values to be substituted into the template. An example would be:
“<STATE>MO</STATE><AUCTIONDATE>1/15/2003</AUCTIONDATE>.”

EB_EMAIL_TEMPLATE – Hold the template_id, subject template, and body template. The subject and body templates include all the text a single tag for each item that is to be replaced. An example would be: “Dear <FIRSTNAME>, Welcome to Equipment Barn!”

All messages are of some “TYPE”, either auction_reminder, or equipmentupdate, etc. Templates also contain a type. This type field is used to match an email to it’s template.

The general process goes as follows:

- 1) One of several scheduled processes (genaucremin.pl or genaualert.pl, etc.) will create emails as necessary and insert them into the email table, these messages will be inserted with a status of “QUEUED”. These scheduled processes are perl scripts that run on the same Windows server machine that runs the webserver....and at this time the database as well although that may be moved off in the future.
- 2) The eb_mailsender.pl will be run on its schedule and will get a list of all “QUEUED” mail ids (guids). These will be retrieved via a webservice call from the Linux box/Qmail server to the EB_MAIL webservice application on the windows server.
- 3) The eb_mailsender will loop through all of the queued mails one by one and retrieve the details such as to, from, etc. Note that the calls here are to another webservice called GetMailDetail.
- 4) The GetMailDetail will get the email detail including the type of mail. Then it will look up the template to be used for the current mail type. Finally it will call a routine in the webservice program named ApplyXMLTemplate. The ApplyXMLTemplate will receive the appropriate Template String and the Value String and the values will be applied to the template. This is done using the Regex (regular expression) class in .Net.
- 5) Finally the GetMailDetail will update the Subject and Text strings in the dataset and return the whole thing through an XML writer to the perl program running on the mailserver.
- 6) Now back in the eb_mailsender program it will inject the mail into the qmail message queue (via the standard perl SMTP Send Mail program) and make another webservice call to change the status of the message from “QUEUED” to “SENT”.
- 7) Finally eb_mailsender will write a line to a log file about messages sent. This completes the email sending process.

Notice that the Linux box never makes a direct connection through the database, it is all done through the webservices.

Finally notice that each of the email are handled one at a time. It would be a very trivial change to make the eb_mailsender.pl get back an xml string containing the details of many (or all) the messages to be sent and do them all at once instead of making calls for each and every message. This is how it was written originally but was changed to simplify logging and make testing easier. Should scalability ever be an issue this would be a very small change.

Hardware Architecture/Infrastructure

There are 3 machines involved \

EBMAIL - A linux machine running Qmail

EBTEST - A windows Windows 2000 Server running both IIS test sites and the test database

EBPROD - A windows Windows 2000 Server running both IIS prod sites and the database

Websites running on Test and Prod box are:

www.equipmentbarn.com - The main production site. (running on EBPROD)

test.equipmentbarn.com - The test site (running on EBTEST)

www.paul-julia.com - My personal production site. (running on EBPROD-on Port 81)

test.paul-julia.com - My Personal test site (running on EBTEST-on Port 81)

ws.equipmentbarn.com - The production webservice listener/application (internal network only)

wstest.equipmentbarn.com - The test webservice listener/application (internal network only)

The Linux (EBMAIL) box is also running an Apache Webserver running www.equipmentbarn.com and www.paul-julia.com, these sites just have basic pages informing the user that the sites are down for maintenace. These will be used by just redirecting the router to the EBMAIL when there is an outage.

The EBTest IIS Server is running on Port 81, this is necessary as my firewall is redirecting all port 80 traffic to the production box. The production box has websites that are called test.xxxx.com-these simply redirect the traffic to the test box websites. The reason it must run them on 81 is that I only have a single IP (which is the root of most of these problems) and redirect goes back to the client and then fails to connect, IIS does not support forwarding (for security reasons) so running it on 81 and having 81 directed to the test box.

There are 3 key environment variables used by the perl scripts and .Net applications.

EB_ENVIRONMENT (TEST,DEV,PROD) - Currently only used to popup an alert to let user know they are in test.

EB_DATABASE (BUTLER1 OR EBPROD or DELL8100) - Used to make the database connection

EB_WEBSERVICE - Used by cron jobs to know the site to connect to in order to run appropriate web services.

(in prod it should be ws.equipmentbarn.com in test it should be wstest.equipmentbarn.com)

Note that ws and wstest are NOT advertised to the outside world-only the intranet.

Scheduled Processes

All of the relating to equipment barn reside in the \Logs folder of the web drive, at the same level as the \apps folder which holds the EquipmentBarn application and EMail webservice folders. All of the jobs currently write out to EBCRON.LOG and are not rotated except manually. If this becomes excessively large it could easily be changed so that each program has it's own log and/or logs were rotated.

At this time all of the scheduled jobs are Perl scripts. The jobs are scheduled via the Windows2000 system scheduler and all write their logs out to a common log file (EBCRON.LOG) in the same folder. Note that the perl scripts use the EB_DATABASE and EB_WEBSERVICE environment variables

The list of scheduled jobs and a brief description of each is as follows:

eb_mailsend.pl - Look for all QUEUED messages in the eb_mail table and send them, mark them as "SENT". This was originally designed to run on the Linux box (before I moved all EB stuff to windows box except actual email handling) and as such makes extensive use of the EB_EMail webservice for all of it's requests and connectivity rather than an ODBC connection.

eb_aucalert.pl -Generate the emails of auctions upcoming in states for which the user has requested to be alerted. The user will be alerted to all auctions posted since the last run of this job.

eb_aucemin.pl -Generate an email 2 weeks prior to auction and again 3 days prior for which the user has requested to be notified.

eb_equalert.pl - Generate the emails of new posted equipment that meets a users specific requests.

Other Jobs

eb_lostpassword - Send email to users who has requested their password. This is not scheduled but is rather "launched" after the lost password record has been created in the database so that this will send it immediately.

The following table lists the jobs to run and their run frequency:

Log Files

The log files associated with the web servers are all located in \Logs folder of the Web drive (E: on both machines). This is the same place for the scheduled process logs. (EBCRON.LOG). When error logging is completed for the applications if possible the logs will reside there as well.

Visio & Code Samples

Load testing results

Security
