Lotus Notes Web-Based File Downloads

By Julian Robichaux, http://www.nsftools.com
originally published on lotususergroup.org, March 2008

Lotus Notes-based web pages – they never quite work the way you think they will, do they? Besides things like basic formatting, authentication, and backend code, one of the things that seems mysterious right out of the box is the HTTP web engine itself. There are so many configuration options in the server document (and elsewhere), and the caching is hard to understand, and even though everything "just works" with the default settings there are any number of tweaks you can make to enhance (or accidentally degrade) performance. But then again, the web server itself is something you can let your friendly neighborhood Admin take care of, right?

Well, it still behooves you as a developer to have some understanding of how the web server works, even if you don't ever touch the configuration. One of the things that has a fairly direct impact on your Domino web pages is how file attachments are handled.

Three Techniques

There are three basic ways you can handle file downloads on a Domino server:

1. As a file in the server's data\domino directory
2. As a File Resource in a database
3. As a file attachment on a document

The first way – files in a server directory – is very straightforward and efficient. You simply copy one or more files to the data\domino directory on the server (or a subdirectory beneath), and those files are available to anyone who hits your website. For example, if you have a file called "LookAtThisFishICaught.jpg" in the data\domino\fish directory of the server, it would be available at http://www.yourserver.com/fish/LookAtThisFishICaught.jpg . This is how default view icons and native Domino Java jar files are served.

While this is an easy way to serve files (and efficient for the Domino server), you also lose some of the nice features of Lotus Notes in doing this. Namely, you can't easily apply an ACL to the files to limit access, and you can't replicate the files from server to server like you can with database-based files.

The second way to store file attachments is as File Resources in a database. In just the same way that you store agents and forms in a database, you can just add a File Resource to a database design and have it available for download. The URL will look something like this: http://www.yourserver.com/yourdb.nsf/LookAtThisFishICaught.jpg . You can optionally add an "?OpenFileResource" command to the end of the URL, but that's not usually necessary.

Using File Resources, you gain the advantage of ACL-based access control as well as replication. This is very handy, and it usually makes File Resources a more attractive option than files in the data\domino directory.

The third way is to attach a file to a document, either in a rich text field (from the Notes client) or on the document itself (from a web file upload). Both of these ways are useful in situations where you want to take advantage of Notes database ACL and replication capabilities and also allow some amount of user (or at least non-Designer) control over adding/updating/deleting the files.

Compression and Decompression

So what does all this have to do with the Domino web server? Well, while these different techniques for storing file attachments have varying degrees of convenience for you [the Notes developer], they also have varying degrees of convenience for your Domino server. Caching is a factor to some degree with smaller files (and would be a great topic of discussion for one of our resident Admins), and so is compression.

The first two methods we discussed above – files in a server directory and File Resources – are very efficient ways for a Domino server to download files. The server grabs a file several bytes at a time and sends those bytes down the wire. Piece of cake.

File attachments on a document can be a little different, though. By default, when you attach a file to a rich text field it gets compressed as it's stored on the document. This makes for a much more efficient way to store the file (because it takes up less space), but it's actually a much LESS efficient way if a Domino server has to serve the file as a download. This is because the server has to *decompress* the entire file before the file can be downloaded.

If you think about this for a moment, it might make sense to you why this is true. When the file is downloaded, it is sent several bytes at a time. However, when an attachment on a document is stored in a compressed state, if the server sends down the bytes as-is then the browser/client that receives the download won't be able to do anything with the file when it receives the entire thing: it'll be a compressed file. Furthermore, the compression algorithm requires the server to decompress the ENTIRE file in order to get the decompressed bytes to download, so it can't do a streaming decompression to get the bytes as it goes. The whole file has to be converted back to it's “normal” form first.

In any case, the whole decompression process doesn't have much impact if your files are small (in fact, small files aren't usually even compressed by Notes in the first place), but if your server is serving up 50 MB Word documents and large PDFs, the decompression process can make a huge difference in download speeds.

To avoid this problem, you can simply deselect the “Compress” option at the time you attach a file. It's a checkbox just under the “Create” button when you select a file attachment, and it's checked by default. Two inconvenient things about this: (1) if you attach a file programmatically or via drag-and-drop you don't have the option to not compress it, and (2) there's no way to do this automatically on all docs in a database, you have to do it doc by doc, attachment by attachment.

While I wouldn't recommend manually going through and changing ALL of your file attachments to remove compression (unless you have a strong need to), it can help with the large file downloads, especially if they're accessed frequently. It can also allow for something

called “byte range serving” for the files, which is good for PDFs that have been saved in “Optimized for web” format so they can be served a page at a time over the wire. Here's a link with more information:

http://www-1.ibm.com/support/docview.wss?uid=swg27003226

GZIP Compression

Even more efficient is to allow the server to send files using Gzip compression, which means the file is compressed by the server as it's being downloaded and decompressed by the client as it's being received. While this does add a tiny (and unnoticeable) CPU load, it can dramatically decrease the amount of bytes that are sent across the wire, which is especially good for slow network connections.

Sadly, as of Domino 8.0.1, native Gzip compression isn't done by the Domino server itself with anything but DWA mail attachments. There are some ways you can force individual files to be served in gzipped format though. I outlined the steps on my blog a few months ago, here:

http://www.nsftools.com/blog/blog-08-2007.htm#08-23-07

In addition, there's at least one third-party tool called Puakma Web Booster that can gzip your Domino content on the fly (with no tricks or changes that you have to do):

http://www.puakma.net/puakma/website.pma/Booster?OpenPage

Gzipping is mainly a factor in serving up large JavaScript libraries, since large files in formats like JPG or PDF are already compressed and can't be compressed much more with gzip, but with the increasing demand for “rich” web applications that use lots of JavaScript, a little planning and understanding can make a difference.