blowery.org

Ben's little playground.

USERLAND

HttpCompressionModule

zipped souce

Make a Donation PayPal Donations Appreciated!

Known Issue: This build will not work with **Server.Transfer!** I'm looking into why it's busted, but no love so far. If anyone has any suggestions, I'd love to hear them!

New version 1.1! Moved up to version 0.31 of SharpZipLib, which should fix a bunch of bugs. More noteably, from my testing, this version seems to fix the problems people were having with compressed images. Last, I reworked the name of the assembly and the way the config sections work, so to use this version, you'll have to make some changes to your application's web.config. Check out the new examples for the correct syntax.

New version! Thanks to Alex Gretha for fixing the filter to work with multiple Response.Write calls! I also fixed up the solution file to not include my crap web project, and updated the sample web.config, and included a readme.txt to get people going.

I'm on this list about all things .net (in the microsoft sense, not the tld sense) and some discussion of HTTP compression cropped up. Apparently, some people were having problems with the ISAPI HTTP compression filter that Microsoft provides. I'd never really looked into HTTP compression before, so I thought I'd see what was involved.

Turns out, it's really pretty simple. HTTP User Agents that support compression will send up a Accept-Encoding header with the kinds of compression they support. Currently, you can send up gzip, deflate, or compress. Interesting. I know that gzip and deflate are implemented by a GPL library, SharpZipLib, so maybe there's a way I can hook into the Http pipeline that microsoft's asp.net provides and compress the content on the way out.

Well, you can. There's a filter property on the HttpResponse object. If you set a Stream into this property, all output will be sent to that stream before being output. This is getting pretty easy all the sudden, especially since SharpZipLib provides deflate

and gzip Stream implementations.

After some experimentation, I figure out that I can't use the compressing stream provided by SharpZipLib directly (for some reason), so I wrote a little stream wrapper around them for HTTP work. Now, the really slick bit. I wrote an IHttpModule implementation that grabs the BeginRequest event from the HttpApplication and sets one of my compressing filters on the Filter propery of the Response object, before most anything else happens. Turns out, I could have sunk most any event as all the content is sent to the filter in one shot, but whatever.

As a last little bit, I wrote up a configuration section section handler for the module to allow a developer to pick a compression level and preferred algorithm to use. Pretty cool, all in all.

If you have comments or patches or anything, lemme know.

Last update: 1/14/2003; 11:26:58 AM.