Virtual Sites and "Host Headers" Explained

Posted At : October 4, 2008 5:20 PM | Posted By : Mark Kruger
Related Categories: Hosting and Networking

Some web developers never bother to learn the nitty gritty stuff that makes up the Internet. I've seen very bright programmers who don't know the difference between a GET request and a POST request (or why they should care). In your journey through the IT landscape it would behoove you to pick up a few tips on *how the web actually works*. In my view you should know the basics of how a web server and browser work together to deliver content. You should know how to setup a web site in IIS or Apache, and you should know when to use a GET and when to use a POST. It also wouldn't hurt to learn about IP addressing, routing, classless subnets, ARP Caching, application pools, JVM Garbage collection, the theory of relativity and the meaning of life.... but I digress.

Among the items I find myself explaining over and over is the concept of a "HOST Header" and how it's used on a web server. Like many of my blog posts this one is intended to help *me* so I can point to it and not have to repeat myself. Now to be fair, this topic is one I sometimes have to cover with customers and site owners who need to know the difference between a dedicated IP address and a "virtual site". Either way, here's a run down of "virtual sites" and "host headers".

The What

What is a virtual site? Simply put, it is a web site that shares an IP address with other sites on the same server. So, for example, if you had a IP address of 1.2.3.4 on a server and you needed 2 sites on it, you could set up DNS to point *both* of them to that IP address and use a "HOST header" to allow the server to differentiate between them. More on that in a moment.

The Why

First, why do we need virtual sites anyway? Well there's an easy answer to that. Most companies do not have an unlimited supply of public Internet IP addresses. At CF Webtools for example, we have about 1 IP address for every 7 or 8 websites we host. Using these addresses we have to support all our servers, all our public devices (routers, firewalls, VPNs, VOIP) and all of our web sites. It would not be possible to do it without virtual sites. Additionally, it is difficult to manage IP addresses and it's easier (from a management point of view) to not have to wrangle with 100 IP addresses on a server supporting 100 web sites. Finally, even though there are 4 billion plus available public IP addresses, the Internet is actually running out of them. Until the hope and dream of IPv6 becomes a reality (when there will be trillions of addresses and even your hamster can get a class B), we have to carefully husband this resource to keep the web growing.

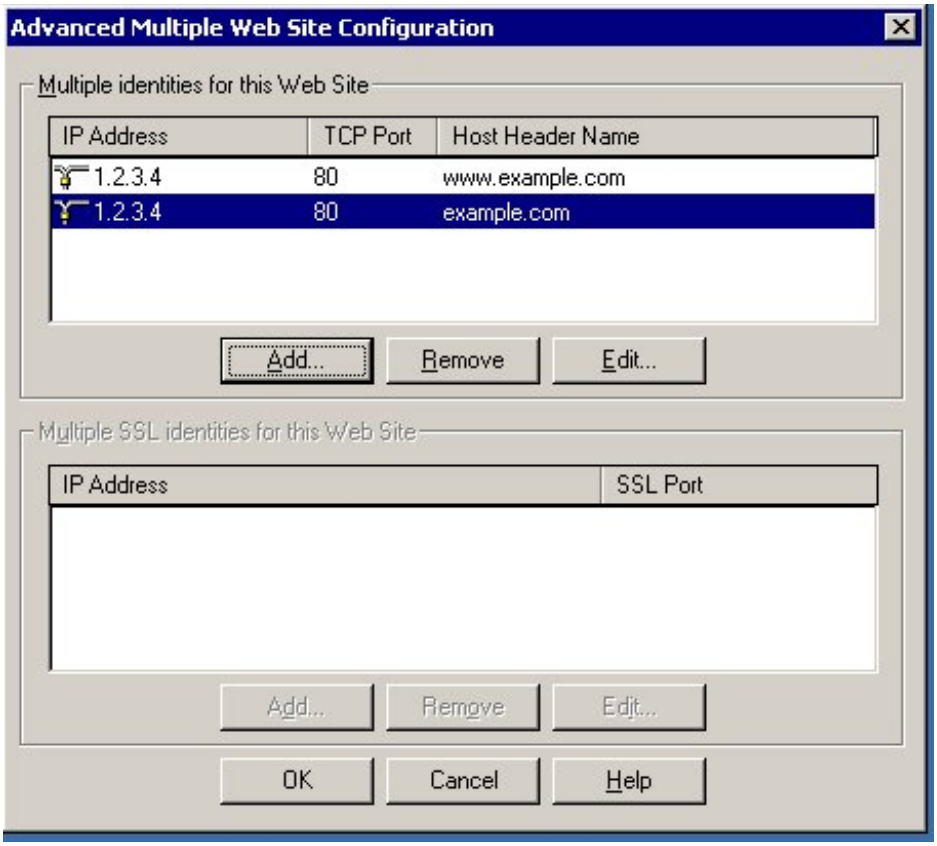
Unpacking an HTTP Requests

First, let's talk about the "HTTP Request". When you type **www.cfwebtools.com** into your browser address bar, the browser does a couple of things (probably more, but here are the tasks we care about today). First, it "looks up" the IP address of the site - in this case 66.37.232.205. So far so good. It knows where to send the request. Next, it constructs an "HTTP Request" to send. This is not as magic as it sounds. It is actually a group of strings separated by line breaks. It looks something like this.

```
GET /index.cfm?blah=true HTTP/1.1

Host: www.cfwebtools.com

Connection: close

User-Agent: Mozilla/5.0 (blah blah)

Accept-Encoding: gzip

Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7

Cache-Control: no

Accept-Language: de,en;q=0.7,en-us;q=0.3

Referer: http://www.coldfusionmuse.com/
```

You might recognize some of that information from the CGI scope in Coldfusion. This is where some of that info comes from (like the user agent and the referrer). Each of these lines after the "GET" command is a "Header". One of them contains the value of "HOST" and it is therefore the "HOST Header". The "Host" is always the domain part of the URL. It's the stuff between the double slashes and the first single slash, as in http://***www.cfwebtools.com***/somefolder/somefile. That's how the phrase "virtual site using a host header" got into our lexicon. The Host header is the key to the web server "figuring out" what to do.

Our hard working little browser (I think I can I think I can) opens a port 80 socket to the IP address for the web site and sends this request through to the web server. Here's where things get interesting. The web server is listening at that IP address on port 80, but it's getting traffic from more than one web site on the same IP address. It unpacks the headers, figures out the host header value, and then looks in it's list for a site to serve based on that value. If it can't find a site with a host header specified that matches, it looks for a site to serve that is set up to listen on that IP *without* a specified host header. In IIS, the host header setup looks like this:



Notice how I need 2 entries, not just one. The second entry (without the www) is for users who simply use "example.com" and leave off the www.

Significance

Why is this important? Well, compared to your first child or the Cubs winning the World Series it's probably not that important. But it does add to your capabilities. Here's one use that I love. I use host headers to create dev sites for customers on our main dev server. I have added a wildcard entry to our DNS for "cfwebtools.com" that points to our dev server. A wildcard entry means *send any host that we can't figure out for this domain to this IP address*. With that in place I can simple open IIS and add a new entry for "anythingIwant.cfwebtools.com" as a host header virtual site and viola! It lives!

When do I Need a Dedicated IP?

Of course there are times when a dedicated IP address is necessary - in particular when you need SSL. If you are going to use a secure Cert (and not use an SSL proxy or load balancer or something) then you will need a dedicated IP address. Why? Because SSL is handled at the site level (that's where the cert lives). The web server cannot read the encrypted host header until it is unencrypted by the certificate - and since it needs to find the site to "know" which certificate to use, the web server and the HTTP request are at a standoff. So instead, certs are bound to IP addresses and the web server hands off to whatever site is listening on the IP address without the host header being specified. In fact, (maybe this has happened to you) if you accidentally leave the "HTTPS" in place in the address bar and then navigate to a site that is actually a virutal site, you may get content that you did not expect from an entirely different, unrelated site. This happens because the browser finds the right IP address, but the web server serves up content from non host header SSL site that is configured for the IP address. Weird huh?

That's the skinny. If you have any tidbits to add feel free to chime in.