Review: BrowserHawk 9

by Steven Smith, Founder of ASPAlliance

Bottom Line Up Front

If you need to detect browser capabilities, browser settings, Java, JavaScript, or Flash, then I would strongly recommend you download cyScape BrowserHawk and give it a test drive. I found the product to be very mature, well-documented, easy to use, and full-featured. What issues I did encounter were "user error" (i.e., my fault), but even still cyScape's support was very responsive and helpful. While researching the market for browser detection controls, I couldn't find any real competitors -- BrowserHawk is the solution unless you want to build your own or live with the very limited capabilities of the ASP.NET Request.Browser object.

My Scenario and Experiences

I've written a custom web advertisement server which currently serves a couple million ad impressions per day. It's important that this application only show certain kinds of ads to users who can support them (for instance, Flash ads to users who have installed Flash), and that our reporting only include non-spider user agents. These are fairly simple requirements, but I also needed these calculations to be accurate, reliable, and extremely fast in order to avoid degrading my site's performance.

Sample Usage - Detect Spiders

One of my primary requirements for this component was that it quickly and accurately detects spiders (or crawlers) so that I could filter these requests from my statistics. While I certainly could have written my own user agent parsing engine to try and capture every known spider, new ones are being written every day (usually stupid ones that don't even know about robots.txt, but that's another issue), and I didn't really want to be saddled with learning about every new spider agent that came along. I was pleased to find that I could implement spider detection in my application with just two lines of code:

Listing 1: Detect Spiders and Crawlers - C#
```
cyScape.BrowserHawk.BrowserObj browObj =
   cyScape.BrowserHawk.BrowserObj.GetBrowser();
bool isCrawler = browObj.Crawler;
```
At this point, it's a simple matter to limit activity logging to only non-crawler user agents, or to terminate the response immediately if the resource in question is not something a crawler should be looking at (for example, the stupid ones that ignore your robots.txt file).

But… is it fast? In short, yes. I did not run it through any vigorous stress tests using thousands of different user agent strings, but I did observe it using ASP.NET's Trace feature as well as run it in production for several months serving upwards of 30 requests per second at times. I never ran into any performance issues as a result of BrowserHawk. (I can say the same for cyScape's CountryHawk, but that is a review for another day.)

For anybody wondering why I didn't just use the built-in ASP.NET Request.Browser.Crawler property, the answer is… I did. However, since the list of browsers this uses is by default rather antiquated and no automatic update service comes with it, I felt it would be best for my users if I invested in a professional and more accurate solution, which BrowserHawk provided. And indeed, there are a lot of user agents caught by BrowserHawk that the default ASP.NET object did not flag as crawlers, so BrowserHawk is in fact more effective.

Sample Usage - Detect Flash

This is covered in the documentation as well. There are two properties in the **ExtendedBrowserObj** object that relate to Flash: **Plugin_Flash** and **Plugin_FlashVerEx**. Normally you'll simply make use of the former. Note that the **ExtendedBrowserObj** must be instantiated early in the page life cycle, before anything has been sent to the client browser (e.g. above your <head> and any Response.Write statements -- typically at the top of Page_Load in ASP.NET apps).

Listing 2: Detect Flash - C#

```
<%
ExtendedOptions options = new ExtendedOptions();
options.AddProperties("Plugin_Flash");
ExtendedBrowserObj extBrow = BrowserObj.GetExtendedBrowser(options);
%>
<html>
Plugin_Flash: <% Response.Write(extBrow.Plugin_Flash); %>
</html>
```

Note that the **ExtendedBrowserObj** is only available in the *Professional* or *Enterprise* edition of BrowserHawk.

That's It?

For my production needs, the two properties above pretty much exceeded my expectations. I had some issues due to my server configuration that were unique to my setup, which caused me some headaches implementing licensing, but cyScape's support was first-class, and they even sent me an updated build within a day or so of my contacting them that corrected the problem for my special case. Now, since this is a review, I don't want to leave you thinking that BrowserHawk only has two useful features. On the contrary -- I actually have quite a TODO list of features I would like to implement with the help of BrowserHawk that will make my sites more user friendly. If you'd like to see what BrowserHawk can tell you about your web visitors, go to their Browser Analysis Page. From there, you'll see everything from your current monitor display settings to your browser plugins to your Internet connection speed.

Additional Cool Features

As you can see if you took a moment to Analyze Your Browser, BrowserHawk can tell you a great deal more than whether the user is a crawler or has Flash installed. Perhaps your site relies on cookies or JavaScript to function properly. Not all browsers support these features, and most allow paranoid users to turn them off. BrowserHawk lets you easily see whether these features are available, allowing you to gracefully degrade (or inform the user they need to turn these features on or upgrade to a modern browser). Perhaps your site includes sensitive information that requires SSL encryption; use BrowserHawk to detect whether the browser supports SSL and if so, what keysize it is using. Otherwise, your users would receive error messages in their browser if your application tries to engage an SSL session when the user's browser has SSL disabled.

Consider the visual design of your site. When the Web was young, design guidelines suggested sites be usable at widths no more than 640 pixels wide. Some years went by, more people bought higher resolution monitors, and these guidelines grew to 800 or even 1000 pixels. Widescreen laptops today ship with 1680 pixel wide screens, but not everybody who comes to your site will have the same size of screen. Detect the user's screen size, installed fonts, and text size, and render your site accordingly.

Plugins are cool browser features that don't always "just work," because not all users have them installed. Flash and Java are the most common browser plugins, but there are many others, including multimedia plugins like QuickTime, RealPlayer, and Media Player. For many entertainment sites, it's important to know which of these plugins the user has installed so the appropriate media format(s) can be presented. Similarly, the user's connection speed can be important when determining whether to send a large media clip or a small one, which is where the Broadband and ConnectionSpeed properties come into play.

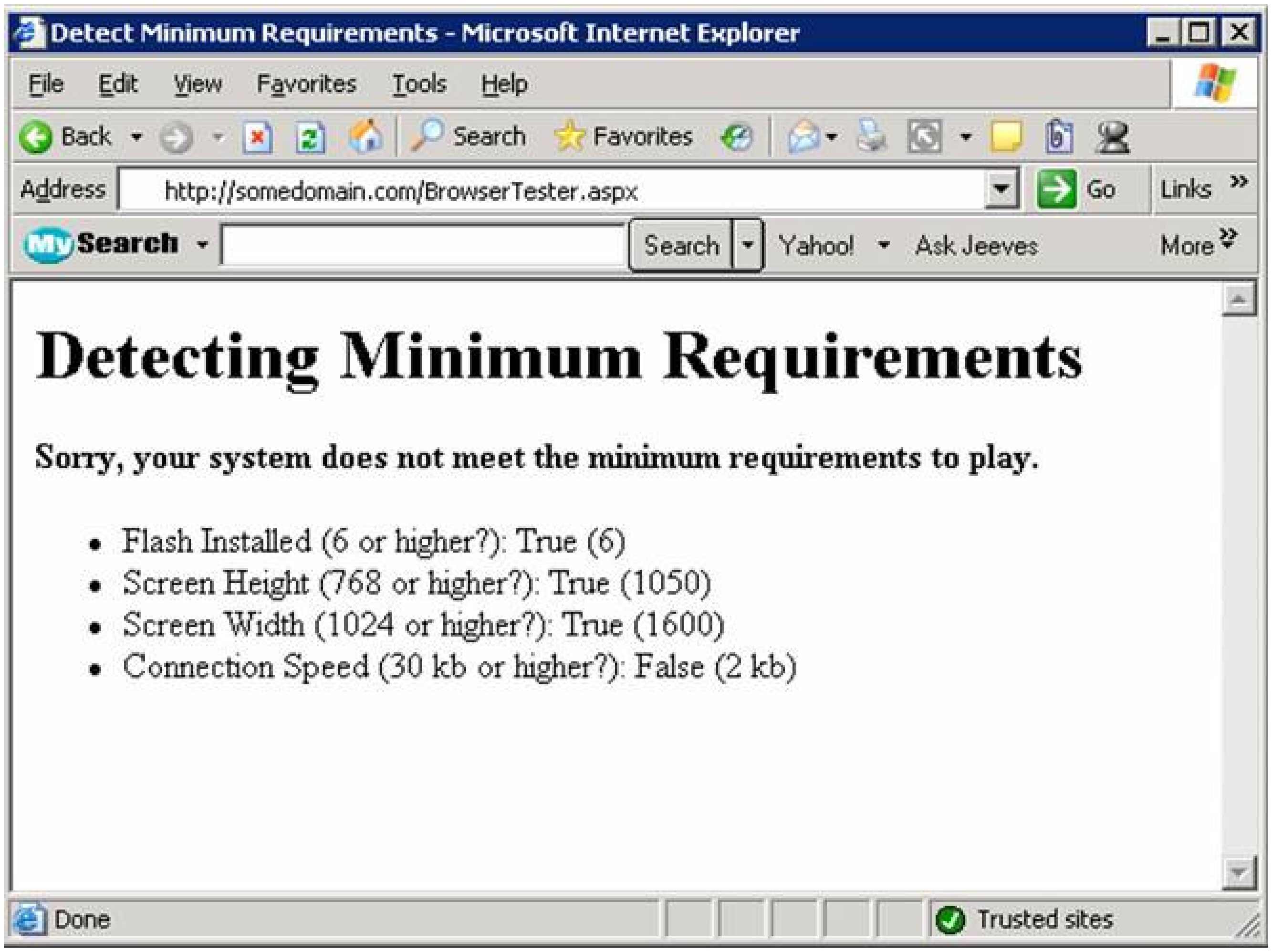
Now, it's all well and good that you can gather this sort of information about individual users, but what if you really want to know a breakdown of all your users' capabilities? Sure, you can write your own script to capture this information and store it in a database or something, but BrowserHawk has already done this for you. Simply configure the component to log statistics (accomplished via a web service on one of your servers), and the rest is done. You're free to create reports against the data, like these (but note that BrowserHawk does not provide a front end for these reports - it just logs the data). According to cyScape's users, it looks like only 0.26% of users are on 640x480 displays, while 12% are still using 800x600. Just about everyone else has at least 1024x768 resolution, with some lucky devils sporting 3840x1024 resolutions.

Going a little deeper into the discussion of web analytics, BrowserHawk's reporting capabilities offer a great deal more depth than most web analysis packages, due to the additional data points it captures from each user. For instance, you can learn how many of your users have broadband versus dial-up, or have cookies or JavaScript enabled. This information can be extremely valuable when you're considering how to architect new features for your site. You can even tie associate session or user ids with the data, so that if a user reports an issue, you can immediately check the database to analyze their browser and system capabilities, which is invaluable for troubleshooting browser-specific problems.

One More Scenario

Let's say you're working on a web site that requires a minimum connection of 30kb/second from your user, a large screen size (1024x768 minimum), and Flash 6 or higher so you can deliver a really cool multiplayer game (or interactive chat, desktop sharing app, or whatever). As part of the application, you test the user's browser and connection speed to determine if they meet your minimum requirements. Doing so with a single C# page would look something like this:

Figure 1: Minimum Requirements Not Met


Detect Minimum Requirements - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://somedomain.com/BrowserTester.aspx Go Links
MySearch Search Yahoo! Ask Jeeves More
Done Trusted sites

Detecting Minimum Requirements

Sorry, your system does not meet the minimum requirements to play.

- Flash Installed (6 or higher?): True (6)
- Screen Height (768 or higher?): True (1050)
- Screen Width (1024 or higher?): True (1600)
- Connection Speed (30 kb or higher?): False (2 kb)

With the requirements met, of course, the site would return a positive message and a link to the game. Listing 3 shows the code required, all in one ASPX page for simplicity.

Listing 3 - Determining Minimum System Requirements Using BrowserHawk

```
<%@ Page language="c#" %>
<%@ Import Namespace="cyScape.BrowserHawk"
%>
<%
ExtendedOptions options = new ExtendedOptions();
options.AddProperties("Plugin_Flash");
options.AddProperties("ConnectionSpeed");
options.AddProperties("Height");
options.AddProperties("Width");
// note these 4 lines can all be done in
// ExtendedOptions constructor in you prefer

ExtendedBrowserObj extBrow = BrowserObj.GetExtendedBrowser(options);
BrowserObj browObj =  BrowserObj.GetBrowser();

bool meetsRequirements = true;

// Detect Flash Settings
int flashVersion = extBrow.Plugin_Flash;
string flashResult = "True (" + flashVersion.ToString() + ")";
if(flashVersion < 6)
{
  flashResult = "False - Please Install Flash";
  meetsRequirements = false;
}

// Detect Screen Size
int height = extBrow.Height;
string heightResult = "True (" + height.ToString() + ")";
if(height < 768)
{
  heightResult = "False - (" + height.ToString() + ")";
  meetsRequirements = false;
}
int width = extBrow.Width;
string widthResult = "True (" + width.ToString() + ")";
if(width < 768)
{
  widthResult = "False - (" + width.ToString() + ")";
  meetsRequirements = false;
}

// detect connection speed
double speed = (double)extBrow.ConnectionSpeed;
string speedResult = "True (" + System.Math.Round((speed/1024),0) + " kb)";
if(speed < 30*1024)
{
  speedResult = "False (" + System.Math.Round((speed/1024),0) + " kb)";
  meetsRequirements = false;
}

string meetsRequirementsMessage = "Your system meets the minimum requirements!";
if(!meetsRequirements)
{
  meetsRequirementsMessage = "Sorry, your system does not meet the minimum requirements to play.";
}
%>
<html><title>Detect Minimum Requirements</title></head>
<body>
<h1>Detecting Minimum Requirements</h1>
<b><%=meetsRequirementsMessage%></b>
<ul>
  <li>Flash Installed (6 or higher?): <%=flashResult %></li>
  <li>Screen Height (768 or higher?): <%=heightResult %></li>
  <li>Screen Width (1024 or higher?): <%=widthResult %></li>
  <li>Connection Speed (30 kb or higher?): <%=speedResult %></li>
</ul>
</body>
</html>
```

Summary

BrowserHawk is a powerful server-side component that integrates into existing classic ASP and ASP.NET sites in just minutes. It is very easy to use, and with just a few lines of code you can detect just about any browser and system settings you need to ensure your web site functions as intended.

BrowserHawk is a very mature product and has been around for nearly a decade now. My real-world testing of the product has shown that it works great. I highly recommend giving it a try and seeing just how much more powerful and elegant BrowserHawk is compared to any home-grown browser sniffer you may be using, and its advantages over ASP.NET's Browser.Request object.

About the Author

Steven Smith is president of AspAlliance.com and AspAdvice.com. He is a Microsoft Regional Director, an ASP.NET MVP, and an ASPInsider Board Member. He is an INETA Speaker Bureau member, and author of two books on ASP.NET.