60 Performance and Optimization Tips for your Website

Learn and implement vital tips on increasing your website s performance. Find out how to make general improvements, how to optimize images, tweak HTML source code, get better results from ASP and ASP.NET, tune your database, and enhance IIS.

Learn and implement vital tips on increasing your website s performance. Find out how to make general improvements, how to optimize images, tweak HTML source code, get better results from ASP and ASP.NET, tune your database, and enhance IIS.

Use one or more suggestions to give website visitors a better impression and keep them on your site longer.

Sections

A. General
B. Images
C. HTML
D. ASP
E. ASP.NET
F. Database
G. IIS

General - Hardware and Monitoring

1. Host your website with a company known for performance. Check newsgroups and web host listing sites such as TopHosts (http://www.tophosts.com) for companies known for high performance. In a shared server environment (not dedicated hosting), find out how many other websites are on the same server as yours. Some web hosts will put up to 1 thousand sites on one server.

2. When hosting your own website, install as much physical RAM as possible on the web server. As memory gets cheaper and cheaper by the minute it is one of the most cost effective purchases an IT department can make.

3. When hosting your own website, use quality hardware ??? don t try and save money when purchasing the hardware that will run one of your most valuable assets.

4. When hosting your own website, consider purchasing quality NIC cards which can offload and reduce CPU usage for networking.

5. Monitor your web pages for performance to make more informed decisions. End-to-end testing of your website, e.g. testing of multiple web pages on your website, is essential to understanding weak points and places for improvement.

Images

6. Reduce the size of images by reducing the number of colors. Optimize your images with online tools like GifWorks (http://www.gifworks.com).

7. Reduce the size of images by reducing its dimensions. Cropping images, also known as relevance-enhancing, can reduce size and also help focus the attention of the quick eye of the web surfer.

8. Use the jpg format for photos. Jpeg compression is known for its exceptional compression abilities with regard to photos. Many tools, such as ACDSee (http://www.acdsee.com/) can easily convert images from one format to another and allow you to set the compression level for jpeg images.

9. Reduce the total number of images on one page. Most web browsers download up to 4 images concurrently; however, each connection adds to the overall response/load time.

10. Reuse images whenever possible to take advantage of server-side (web server) and client-side (browser) caching.

11. Specify image WIDTH and HEIGHT attributes for faster loading in the browser.

12. Combine images and use image maps instead of segmenting images, remembering that each image loaded is a connection between the web browser and the web server.

13. Use image ALT attributes sparingly. Each character adds to the size of the web page and to the load time.

HTML

14. Use comments sparingly. While invisible when a web page is loaded in a browser, the comments are still transferred from the web server, needlessly wasting bandwidth.

15. Avoid unnecessary HTML tags: not all tags need a closing tag. For example, it is not necessary to have a close or tag.

16. Optimize your HTML code with free online tools such as iWebTool (http://www.iwebtool.com/html optimizer) to reduce the overall size of the html file.

17. Avoid using frames. Each frame loads its own web page which can increase the overall response time of a webpage.

18. Minimize the amount of text and sub-tags between the HEAD open and close tag.

19. Place external JavaScript tags, i.e. with src=??? someurl ???, at the end of the document to delay loading.

20. Simplify tables and avoid nested tables (tables within tables).

21. In tables use background colors instead of images.

22. Avoid using WYSIWYG editors, e.g. FrontPage, that include extra and irrelevant text and HTML tags. Learn to edit HTML code by hand.

ASP

23. Disable Session State if not using sessions. If you are using sessions, consider using cookies or an id in the query string along with temporarily storing data in a database. To disable session state, at the top of your ASP page, include the directive:

24. Use Option Explicit to reduce coding errors. At the top of your ASP page, include the directive:

25. Use Server.Transfer over Response.Redirect. Response.Redirect uses an additional round-trip to the web server whereas Server.Transfer does not, reducing the amount of bandwidth used, system resources, and overall response time.

26. When specifying a URL without a web page, e.g. http://www.exclamationsoft.com, include the trailing slash, e.g. http://www.exclamationsoft.com/, to save a trip back to the web server.

27. Reduce the use of global variables.

28. Reduce the number of include files used on a page. Also, segment and categorize functions in commonly used include files.

29. Be careful of string concatenation as the size of the string grows. As the string grows through concatenation, it is copied to a new location in memory each time.

30. Set objects, especially database objects to Nothing when no longer needed. For example, Set oRecordset = Nothing.

31. Keep blocks of ASP script together. Each switch between ASP script and HTML causes the compiler to stop and start processing.

32. Don t use ASP commenting: which is compiled each time the page is loaded. Instead, use HTML commenting or no commenting at all.

33. Don t leave empty Session OnStart or Session OnEnd methods. If Sessions are not used in your application, remove these two methods since they will be compiled and executed even when empty.

ASP.NET

34. ASP.NET Cache API. If you are not using the Cache API, stop your coding and read the help on this subject and implement it as soon as possible.

35. Reduce multiple database resultsets. Each database query with returned results is a round trip to the database server, adding to the overall response time.

36. Use "paged" data access, e.g. ASP.NET makes it easy to create DataGrids and DataLists of results from database queries. Use them to your advantage by only showing a small sub-set of those results to improve web page response times.

37. Use HttpContext.Items to add frequently used objects during a single page load to create a "per-request" cache.

38. Utilize background processing to help with long running tasks. Create multi-threaded operations when possible and feasible.

39. Use quick page caching for pages that are displayed repeatedly (think auto-refresh):

40. If not using form post-back, turn off viewstate:

41. Avoid catching unnecessary exceptions: exceptions incur a large amount of overhead and should not be used for program logic flow control.

42. Avoid throwing exceptions.

43. Enable buffering.

44. Use Page.IsPostBack to prevent code from being needlessly executed.

45. Ensure debug is set to false and the release build is used in production.

46. Use client-side validation controls to reduce server round-trips.

47. Use StringBuilder when concatenating strings.

Database

48. Use SQL Server or another professional level database over Access.

49. Use stored procedures over simple SQL queries.

50. Connection Pooling - ensure connection pooling is enabled by using a DSN (Data Source Name) in the ODBC Data Source Administrator.

IIS

51. Use server side compression software such as Port80???s httpZip (http://www.port80software.com/products/httpzip/)

52. Use GZip compression to reduce bandwidth, but be aware that CPU utilization may go up. A-B testing can help you find the right balance of whether GZip compression is right for your website.

53. Use IIS 6.0, which includes significant performance enhancements such as Kernel Caching.

54. Don t install or use Microsoft Index Server unless you need it. The cost far outweighs the benefits that Index Server will give.

55. Don t enable logging unless needed. Logging is disk and resource intensive.

56. Run IIS "in-process" if the website is mostly static. Mostly dynamic or unstable websites should not use this setting.

57. Enable "Cache ISAPI applications" for sites that use ASP which benefits greatly from this setting.

58. Disable debugging on production servers.

59. Enable "HTTP Keep Alives" for IIS 5.0. This setting is enabled by default in IIS 6.0.

60. Shorten connection timeouts to reduce the hold on resources.

Resources used:

1. MSDN
2. Chapter 6 - Improving ASP.NET Performance
3. Tips to Improve ASP Application Performance
4. IIS 101: The Basics of Performance Tuning
5. Top Ten Ways To Pump Up IIS Performance

Copyright 2006 ExclamationSoft

Short note about the author

Tim Hodgson manages software development for ExclamationSoft (http://www.ExclamationSoft.com). He is an expert on website and server performance and availability monitoring and writes articles on a wide variety of computer related topics.

Author: Tim Hodgson
Article downloaded from page **eioba.com**